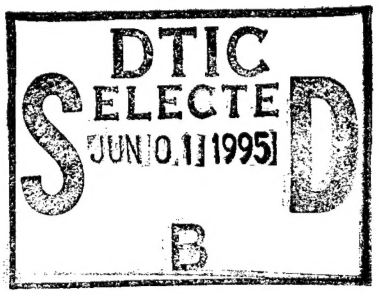
Technical Report GL-95-3
April 1995

US Army Corps of Engineers
Waterways Experiment Station

Backcalculation of Flexible Pavement Moduli from Falling Weight Deflectometer Data Using Artificial Neural Networks

by Roger W. Meier



Prepared for Headquarters, U.S. Army Corps of Engineers

Technical Report GL-95-3
April 1995

Backcalculation of Flexible Pavement Moduli from Falling Weight Deflectometer Data Using Artificial Neural Networks

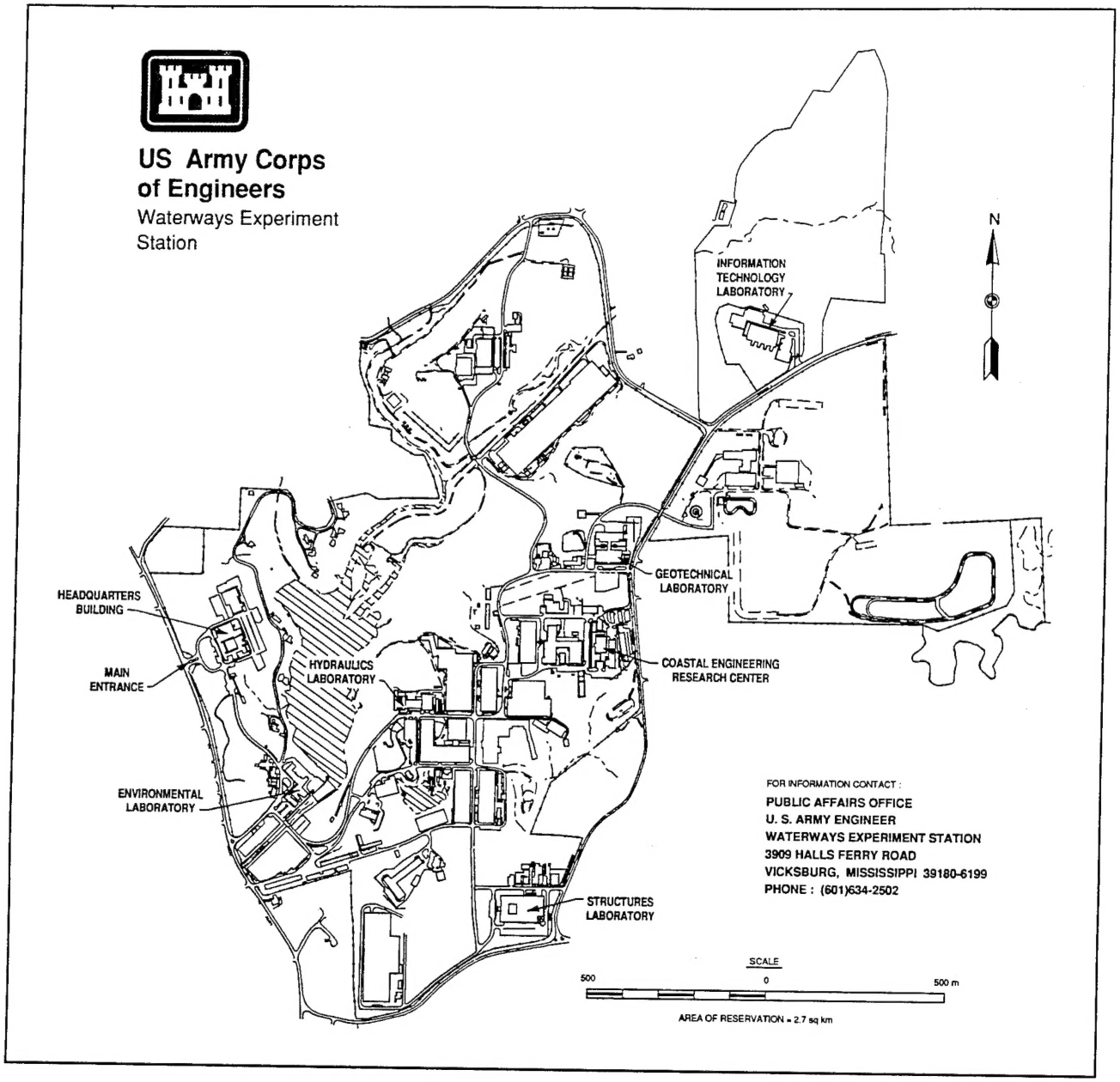
by Roger W. Meier

U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final report

Approved for public release; distribution is unlimited

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000



Waterways Experiment Station Cataloging-In-Publication Data

Meier, Roger W.

Backcalculation of flexible pavement moduli from falling weight deflectometer data using artificial neural networks / by Roger W. Meier ; prepared for U.S. Army Corps of Engineers.

239 p. : ill. ; 28 cm. — (Technical report ; GL-95-3)

Includes bibliographic references.

1. Pavements, Flexible. 2. Non-destructive testing. 3. Neural networks (Computer science) I. United States. Army. Corps of Engineers. II. U.S. Army Engineer Waterways Experiment Station. III. Geotechnical Laboratory (U.S. Army Engineer Waterways Experiment Station) IV. Title. V. Series: Technical report (U.S. Army Engineer Waterways Experiment Station) ; GL-95-3.

TA7 W34 no.GL-95-3

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PHOTOGRAPHS

This research was conducted during the period from October 1993 through September 1994. It was funded through the Laboratory Discretionary Research and Development program at the U. S. Army Engineer Waterways Experiment Station (WES). The computational work was performed using the resources of the High Performance Computing Center at WES.

This research was conducted under the general supervision of Dr. William F. Marcuson III, Director, Geotechnical Laboratory (GL), and under the direct supervision of Mr. Newell R. Murphy, Jr., Chief, Mobility Systems Division (MSD), GL, and Mr. Donald D. Randolph, Chief, Modeling Branch, MSD. Special thanks are due Dr. Albert J. Bush III, Pavement Systems Division, GL, for his help in focusing this research and for his invaluable advice throughout the study.

This report was prepared by Dr. Meier as partial fulfillment of the requirements for the Doctor of Philosophy degree in Civil Engineering at the Georgia Institute of Technology. Dr. Meier is particularly indebted to his dissertation advisor, Dr. Glenn J. Rix, for the friendship, support, and guidance provided throughout his doctoral studies.

At the time of publication of this report, Dr. Robert W. Whalin was the Director of WES and COL Bruce K. Howard, EN was the Commander.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products.

SUMMARY

The Falling Weight Deflectometer (FWD) test is one of the most widely used tests for assessing the structural integrity of pavement systems in a nondestructive manner. A major limitation of existing techniques for backcalculating pavement layer moduli from FWD results is that they are computationally inefficient. This not only makes them tedious to use, it also constrains them to employ simplified static models of the FWD test that can be computed relatively quickly. Studies have shown that significant errors in the backcalculated pavement moduli can accrue from using a static model of what is inherently a dynamic test.

The goal of this research was to develop a method for backcalculating pavement layer moduli from FWD data in real time. This was accomplished by training an artificial neural network to approximate the backcalculation function using large volumes of synthetic test data generated by static and dynamic pavement response models. One neural network was trained using synthetic test data generated by the same static, layered-elastic model used in the conventional backcalculation program WESDEF. That neural network was shown to be just as accurate but 2500 times faster. The same neural network

was subsequently retrained using data generated by a elastodynamic model of the FWD test. The dynamic analysis provides a much better approximation of the actual test conditions and avoids problems inherent in the static analysis. Based on the amounts of time needed to create the static and dynamic training sets, a conventional program would likely run 20 times slower if it employed the dynamic model. The processing time of the neural network, on the other hand, is unchanged because it was simply retrained using different data.

These artificial neural networks provide the real-time backcalculation capabilities needed for more thorough, more frequent, and more cost-effective pavement evaluations. Furthermore, they permit the use of more-realistic models, which can increase the accuracy of the backcalculated moduli.

CONVERSION FACTORS, NON-SI TO SI (METRIC) UNITS

Non-SI units of measurement used in this report can be converted to SI units as follows:

Multiply	By	To Obtain
feet	0.3048	meters
inches	2.54	centimeters
kips	4.448222	kilonewtons
miles per hour	0.44704	meters per second
mils	25.4	micrometers
pounds (force)	4.448222	newtons
pounds (force) per square inch	6.894757	kilopascals

CHAPTER 1

INTRODUCTION

Background

The structural integrity of roads and airfields is determined in large part by the load-deflection properties of the concrete, asphalt, and soils that make up the pavement system. Those properties can be measured in the laboratory; however, doing so requires that a portion of the pavement system be destroyed in order to obtain material samples. The disturbance that results from the sampling process itself calls into question the accuracy of the subsequent laboratory tests. Furthermore, the inherent heterogeneity of pavement materials (especially in the base and subgrade) means that isolated samples are often not representative of the pavement system as a whole.

As an alternative to laboratory testing, the structural properties of the pavement can be measured *in situ* using techniques that fall under the general category of *nondestructive testing* (NDT). As the name implies, NDT can be used to assess pavement fitness without destroying the pavement in the process. Unlike laboratory methods, there are no concerns with sample

disturbance because samples are not taken. Furthermore, the test results reflect the properties of the pavement system over a broad area rather than at a single point. In fact, to the extent that the device being used simulates an actual vehicle load, the NDT techniques can actually represent full-scale tests of the pavement system.

One of the most widely used NDT techniques is the Falling Weight Deflectometer (FWD) test. An FWD test is performed by applying an impulse load to the pavement via a circular plate and measuring the resulting pavement deflections directly beneath the plate and at several radial offsets from the plate. The experimental data is generally summarized as a *deflection basin* that is constructed from the peak deflections recorded at each of the measurement locations. The stiffnesses of the various material layers in the pavement system are calculated from these deflection basins through a process called *backcalculation* or *inversion*.

Objectives

A major limitation of existing techniques for backcalculating pavement layer moduli from FWD results is that they are computationally inefficient—they all involve numerous repetitions of mathematically-complex calculations. This makes the inversion programs slow and tedious to use. Though the test itself takes only a minute or two to run, the data analysis can take considerably longer. This can limit the usefulness of FWD testing, especially for performing

routine, periodic assessments of pavement integrity such as would be needed for the FWD to be used as part of a pavement management system. This research is aimed at eliminating the inversion bottleneck by using artificial neural networks to expeditiously backcalculate pavement layer moduli from FWD deflection basins. The real-time backcalculation afforded by artificial neural networks will not only provide increased productivity and permit an increased frequency of testing, it also opens up the possibility of developing NDT techniques that both measure and evaluate pavement properties in real time. Real-time NDT would not only allow more pavement to be tested more often, it would also reduce the costs of performing each test because it would no longer be necessary to close a traffic lane and divert traffic. That alleviates both the direct costs of traffic control and the indirect costs of commuting delays as well as increasing the productivity of the test crews.

Artificial neural networks have their origins in research on the behavior of the human mind. They were developed to reduce mental processes—such as learning, reasoning, and memory—to mathematical abstractions in hopes of better understanding them. Artificial neural networks are collections of highly-interconnected but mathematically-simple processing elements (usually implemented in a digital computer program) that exhibit certain brain-like traits such as learning by example and generalizing from imperfect examples. In the context of FWD backcalculation, a neural network can be "taught" to map

deflection basins onto their corresponding pavement layer moduli by repeatedly showing it examples of the correct mapping. This training could be accomplished using experimentally-determined deflection basins and pavement layer moduli. Alternatively, synthetic deflection basins obtained by running a computer model of the FWD test can be used. The latter approach, which produces a backcalculation neural network functionally identical to existing basin-matching backcalculation programs, was taken in this study.

This research can be logically separated into two parts. The goal of the first phase of the research was to show that it is possible to perform real-time backcalculation of pavement layer moduli using artificial neural networks. With that in mind, an artificial neural network was trained using synthetic deflection basins generated by a computer program that implements a static analysis of pavement response. That same static analysis is incorporated in several conventional backcalculation programs that are widely used. This allowed a direct comparison of both speed and accuracy between the artificial neural network and conventional approaches.

The goal of the second phase of the research was to enhance the accuracy of the neural network by retraining it with synthetic deflection basins developed using an elastodynamic analysis of the pavement response. The dynamic analysis provides a much better approximation of the actual FWD test conditions. It also avoids certain pathologies, such as an excessive sensitivity

to the assumed bedrock depth, that are inherent in the static analysis. Because the computational efficiency of a trained neural network is completely independent of the computational complexity of the program used to generate its training set, it is possible to train a network to account for the dynamics of the FWD test without increasing its processing time. Undeniably, it will take significantly longer to create the training set for the neural network. Once trained, however, the network will be able to backcalculate moduli just as quickly as one trained using a static solution. Contrast this with a conventional gradient search program: the conventional program must repeatedly solve the more-complex dynamic problem to obtain a solution, resulting in tremendous increases in computation times.

Organization

An overview of the devices most commonly used for nondestructive testing of pavements is presented in Chapter 2 along with a summary of the traditional methods used to backcalculate pavement layer moduli from the experimental results. Particular emphasis is placed on the computational inefficiencies of existing methods and the problems that may accrue from performing a static analysis of dynamic data. The concept of inversion using artificial neural networks is introduced there as well.

Chapter 3 provides an historical and mathematical background for the multi-layer, feed-forward artificial neural networks employed in this research.

The algorithms needed to implement that type of artificial neural network are presented, as are the algorithms needed to train the network using the error backpropagation technique.

Chapter 4 describes the generation of the neural network training set using a conventional static analysis of pavement response. The underlying assumptions and mathematical details of the static model are presented first. The pavement profile assumptions used to constrain the training set to a reasonable size are discussed next. The chapter concludes with a description of the overall design of the training set itself.

Chapter 5 describes the generation of the neural network training set using an elastodynamic analysis of pavement response based on Green functions. The mathematical details of the analysis technique are presented first. Next, the techniques used to discretize the pavement system are discussed. Finally, the development of a mathematical analogue for the FWD loading pulse is described.

The training and testing of the artificial neural networks are described in Chapter 6. Considerations of the network architecture are discussed as is a method for making the neural networks more robust against the noisy deflection basins typical of real-world experimental data. Also included are the results of a head-to-head comparison of speed and accuracy between the neural network trained using deflection basins generated with a static model of

the FWD test and a conventional backpropagation program incorporating the same static model. Finally, Chapter 6 discusses the retraining of the original neural network using the deflection basins generated using a dynamic model of the FWD test that is more representative of the test conditions.

The research program is summarized in Chapter 7. Conclusions drawn from this study and recommendations for future research are also presented. Suggestions for similar applications of neural network technology to other types of experimental data inversion are also given.

Finally, Appendix A contains the source code listing for the computer program used to train and test the artificial neural networks.

CHAPTER 2

NON-DESTRUCTIVE TESTING OF PAVEMENTS

The majority of the NDT methods currently used for the structural evaluation of pavement systems are based on the same general principle: the structural integrity of a pavement system is inversely proportional to the amount of surface deflection observed under a given load[1]. The primary differences between the various devices being used lies in the nature of the loads they apply to the pavement. Accordingly, NDT devices can be categorized according to load type as being either static, steady-state dynamic, or transient dynamic devices. Examples of the various types are described in the next three

---

[1] The exception to this rule is the Spectral Analysis of Surface Waves method (Nazarian and Stokoe, 1989) and its predecessor, the Surface Wave Method (Jones, Thrower, and Gatfield, 1968). These methods use the propagation velocity of Rayleigh waves to determine the pavement layer shear moduli. Unlike the other NDT methods, which induce stresses and strains in the pavement that are of the same order of magnitude as those imposed by actual traffic, these methods measure the response of the pavement system to the infinitesimal strains produced by the propagation of low-amplitude seismic surface waves.

sections. Additional details can be found in Smith and Lytton (1985), Hoffman and Thompson (1982), and Bentsen, Bush and Harrison (1989).

Static NDT Devices

Static devices measure the deflection response of the pavement to what is essentially a static vertical load. This class of NDT devices includes the Benkelman Beam and the La Croix Deflectograph.

The Benkelman Beam was one of the earliest NDT devices used on pavements. In principle, it is nothing more than a 12-ft beam pivoted at its third point (Figure 1). The applied load is usually provided by a two-axle truck with dual rear wheels. The beam is oriented in the direction of vehicle motion with the probe end placed between one pair of the dual rear tires. The opposite end of the beam moves a dial gauge that records the deflection of the pavement. The deflections are usually measured as the truck moves away from the beam at an extremely slow speed. (In that respect, "quasi-static" would be a more rigorous description of these devices because the load at the measurement point slowly decreases to zero as the truck pulls away.)

Though relatively inexpensive and easy to use, the Benkelman Beam suffers from a number of problems. One problem is that it provides very limited information—the pavement deflection is only measured at one point and frequently only the maximum deflection is recorded. Another problem is that it measures the pavement's response to what is essentially a static load. It is

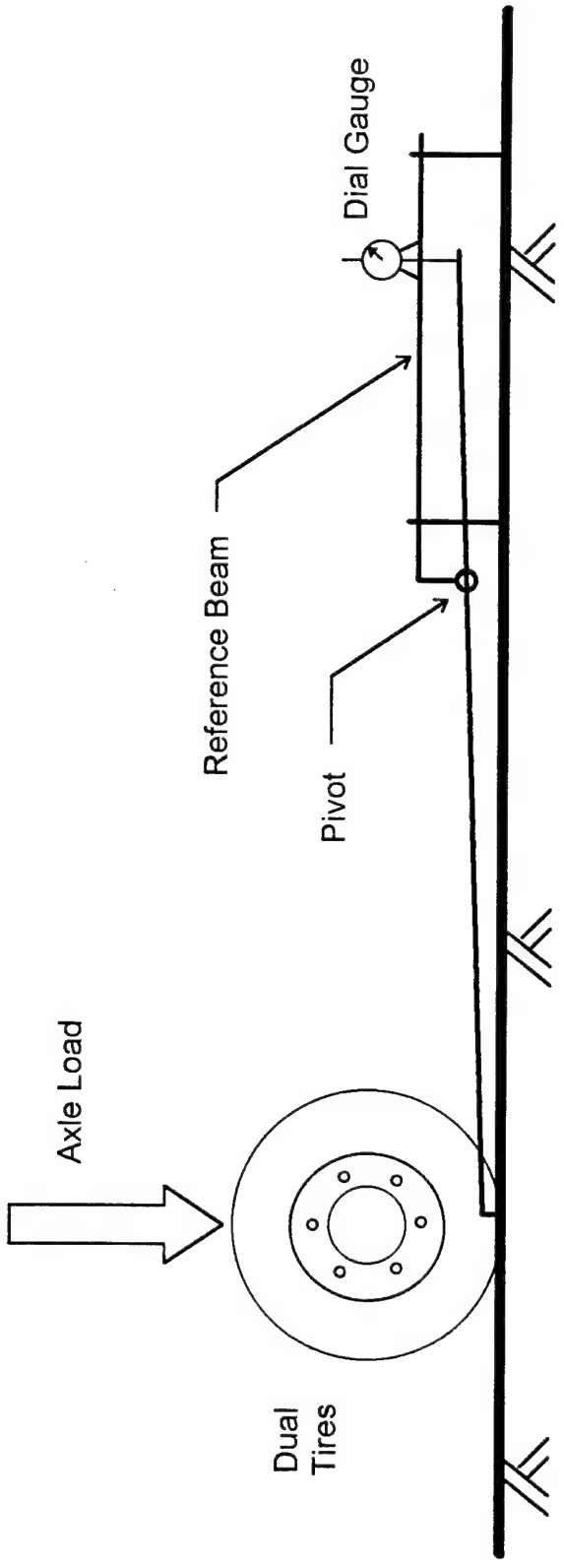


Figure 1. Schematic of the Benkelman Beam

difficult to extrapolate the experimental results to vehicles moving at highway speeds. Finally, it is difficult to ensure that the pivot point of the beam lies outside the deflection basin (Figure 2). On pavements underlain by relatively soft subgrades, the radius of the deflection basin often exceeds the 8-ft distance between the probe end and the pivot point. This means the pivot point is initially below the undeformed elevation of the pavement surface. As the truck pulls away, the pavement rebounds and the elevation of the pivot point changes. This lack of a stable zero reference can, at best, render the results meaningless. At worst, if the problem is not recognized, it can result in an underestimation of the pavement deflection and an overestimation of the structural integrity of the pavement system.

The La Croix Deflectograph is a semi-automated version of the Benkelman Beam. The La Croix Deflectograph consists of two beams (one for each pair of dual rear wheels) mounted on a moveable frame that is suspended beneath the truck supplying the loads. During a test, the frame is positioned on the pavement surface with the probe ends of the beams aligned between each pair of dual rear wheels. The deflection of the pavement is measured as the rear wheels of the truck approach the stationary frame. The frame is then lifted from the pavement and repositioned for the next test. Because the La Croix Deflectograph is based on the Benkelman Beam, it suffers from exactly the same problems.

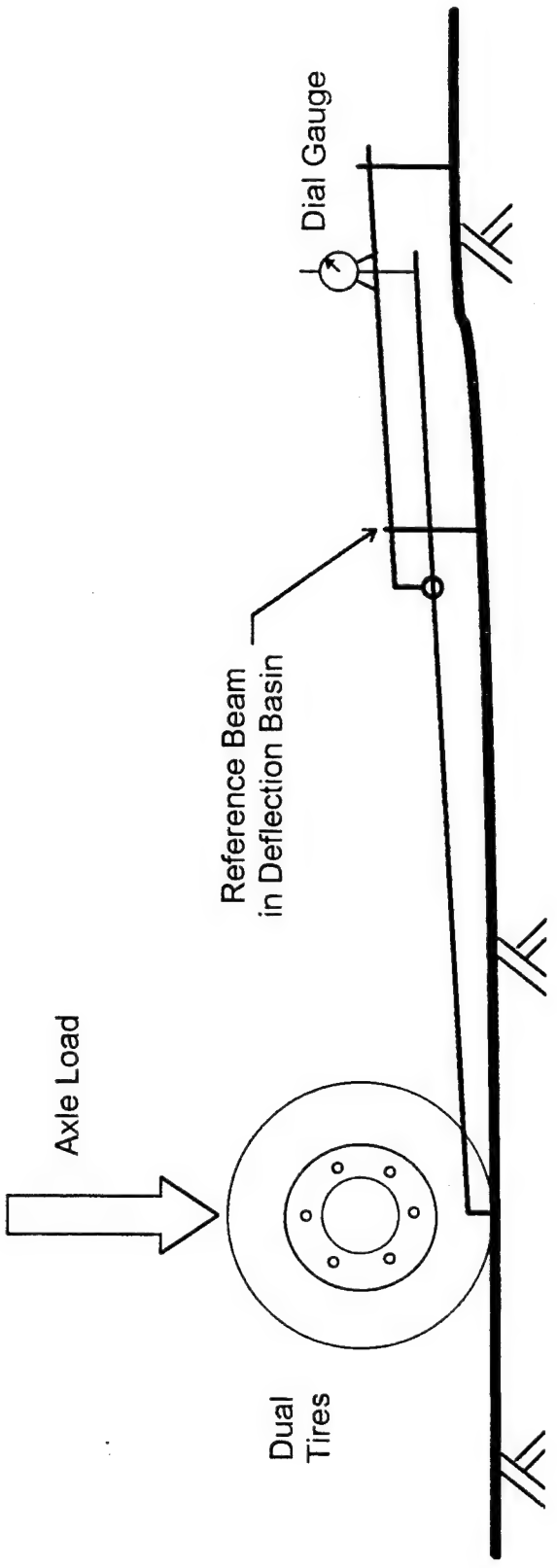


Figure 2. A source of measurement error when using the Benkelman Beam

Steady-State Dynamic NDT Devices

Whereas the static devices described in the preceding section use an actual truck to load the pavement, the dynamic NDT devices strive to duplicate those same loads without driving an actual vehicle over the pavement. In the United States, a common highway design criterion is an 18-kip single axle load (Yoder and Witczak, 1975). Assuming two wheels per axle, a 9,000-lb wheel load would have to be applied in order to duplicate those design conditions. Furthermore, in order to simulate the passage of a vehicle, the load would have to start from zero, rise to a peak, and return to zero. The time period within which this rise and fall would have to occur depends on the velocity of the vehicle. Barksdale (1971) showed that the vertical stresses near the surface of the pavement could be approximated by a half sinusoid with a pulse width that varies as the inverse of the vehicle speed (Figure 3). At a speed of 45 mph, Barksdale calculated a pulse width of approximately 20 msec. This would correspond to a load oscillating at a frequency of approximately 25 Hz.

The NDT devices that fall into the steady-state dynamic category measure the deflection response of the pavement to a low-frequency oscillatory load. The Dynaflect, Road Rater, and WES 16-kip Vibrator are three such steady-state devices.

The Dynaflect (Photograph 1) was one of the earliest steady-state NDT devices. It is a self-contained, trailer-mounted device that can be towed by

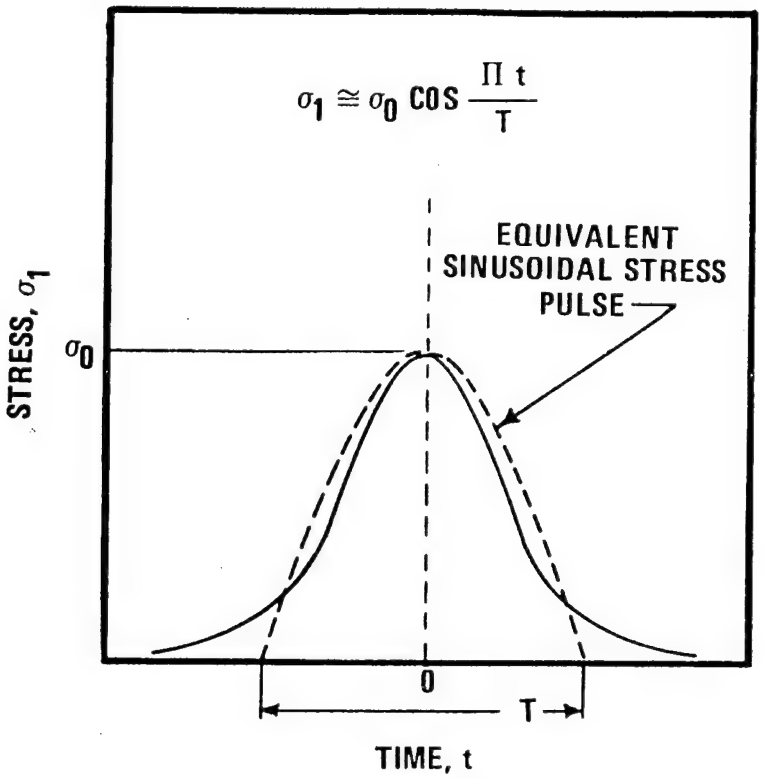
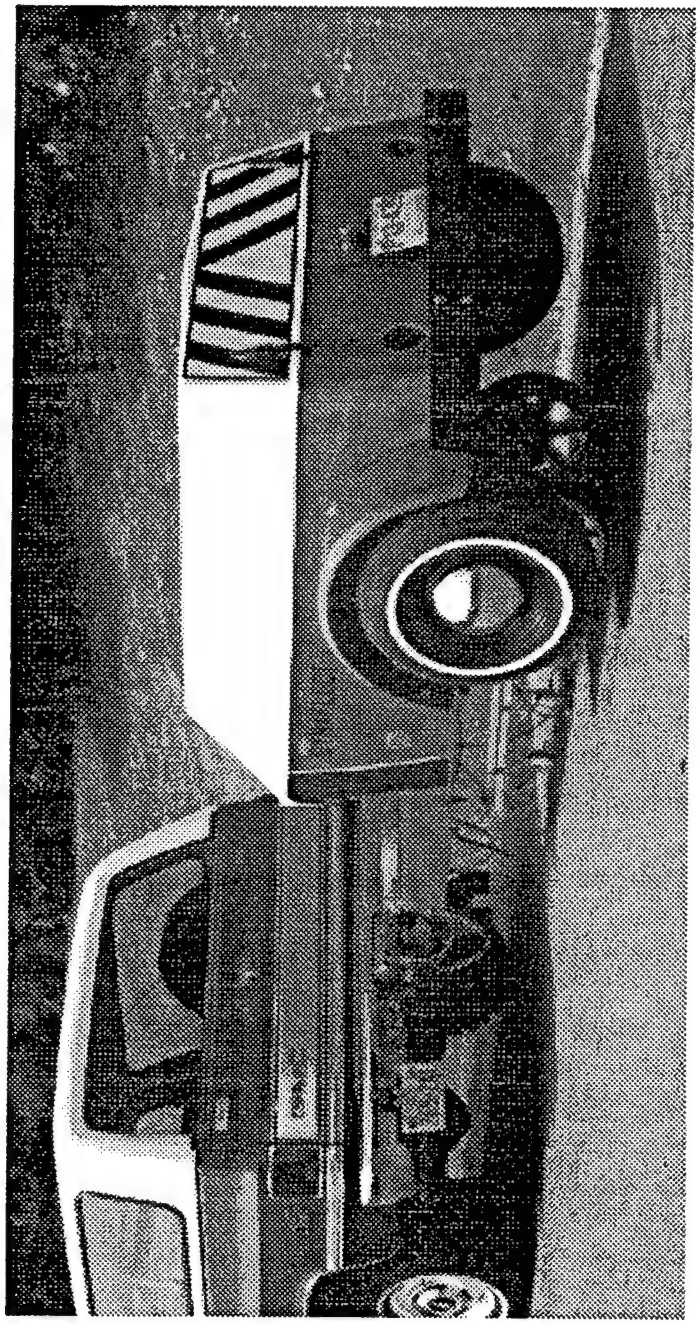


Figure 3. Sinusoidal approximation of load applied by a moving wheel (Barksdale, 1971)



Photograph 1. The Dynaflect

conventional vehicles. The Dynaflect uses a pair of eccentric counter-rotating masses to apply a harmonic load to the pavement. The load oscillates at a frequency of 8 Hz. The harmonic load has a peak-to-peak amplitude of 1000 lb that is superimposed on a static load of 2000 lb (the weight of the device). The actual load therefore varies between 1500 lb and 2500 lb (Figure 4). The load is applied to the pavement through a pair of polyurethane-coated steel loading wheels spaced 20 in apart. Each loading wheel has a diameter of 16 in a width of 4 in. The surface deflections of the pavement are measured with a series of five geophones (velocity transducers). The first geophone is positioned directly between the loading wheels and the remainder are spaced at one foot intervals. The surface velocity records are time integrated by on-board circuitry to produce deflection outputs.

The Road Rater (Photograph 2) is also a self-contained, trailer-mounted device. It uses a hydraulic actuator to apply the static load to the pavement surface. The trailer itself is used as the reaction mass for the hydraulic actuator, which limits the static load to the weight of the trailer. The superimposed harmonic load is generated by a lead-filled steel mass that is accelerated by a servo-controlled actuator. Unlike the Dynaflect, both the amplitude and the frequency of the harmonic load as well as the magnitude of the static load can be varied. The load ranges vary from model to model. The largest model has an 8000-lb reaction mass and can generate a harmonic load

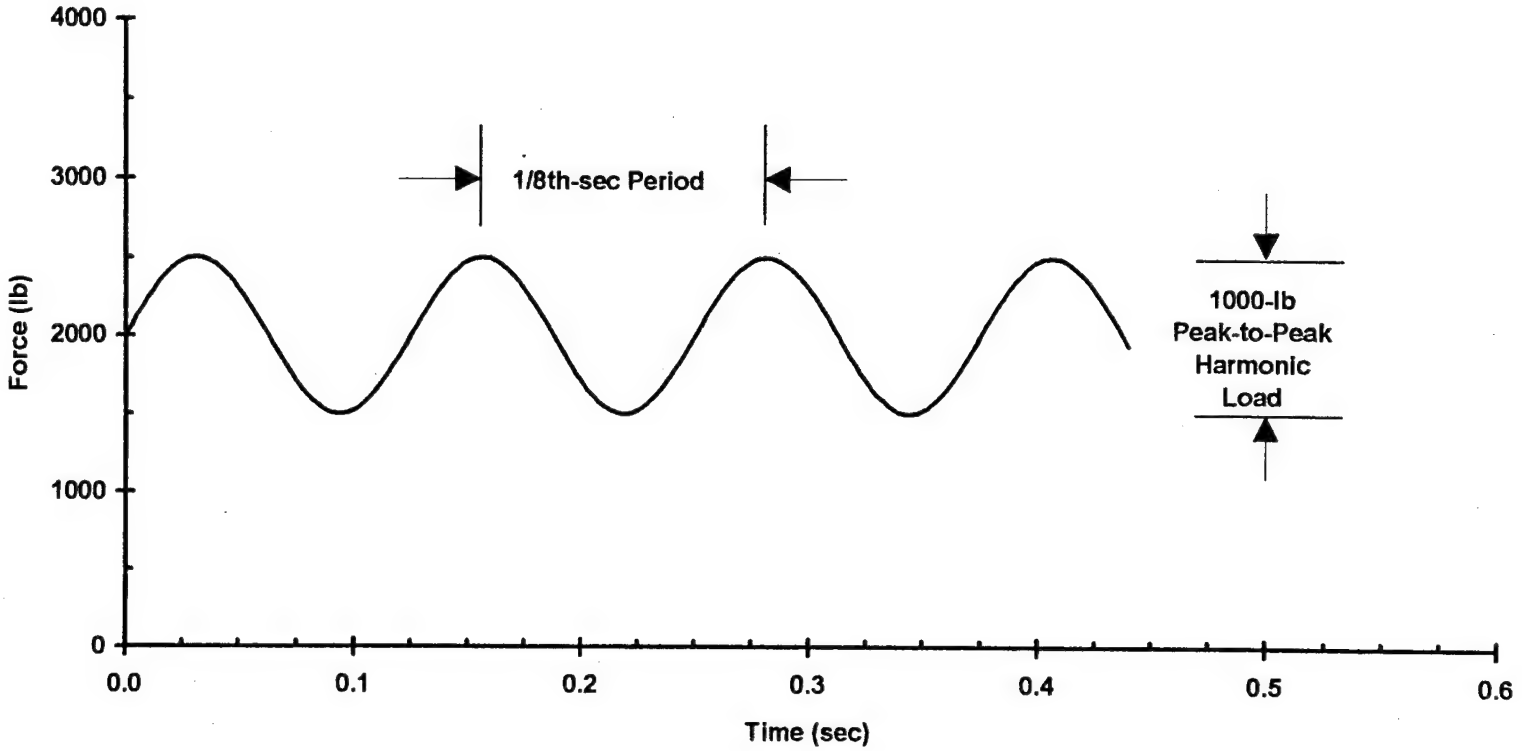
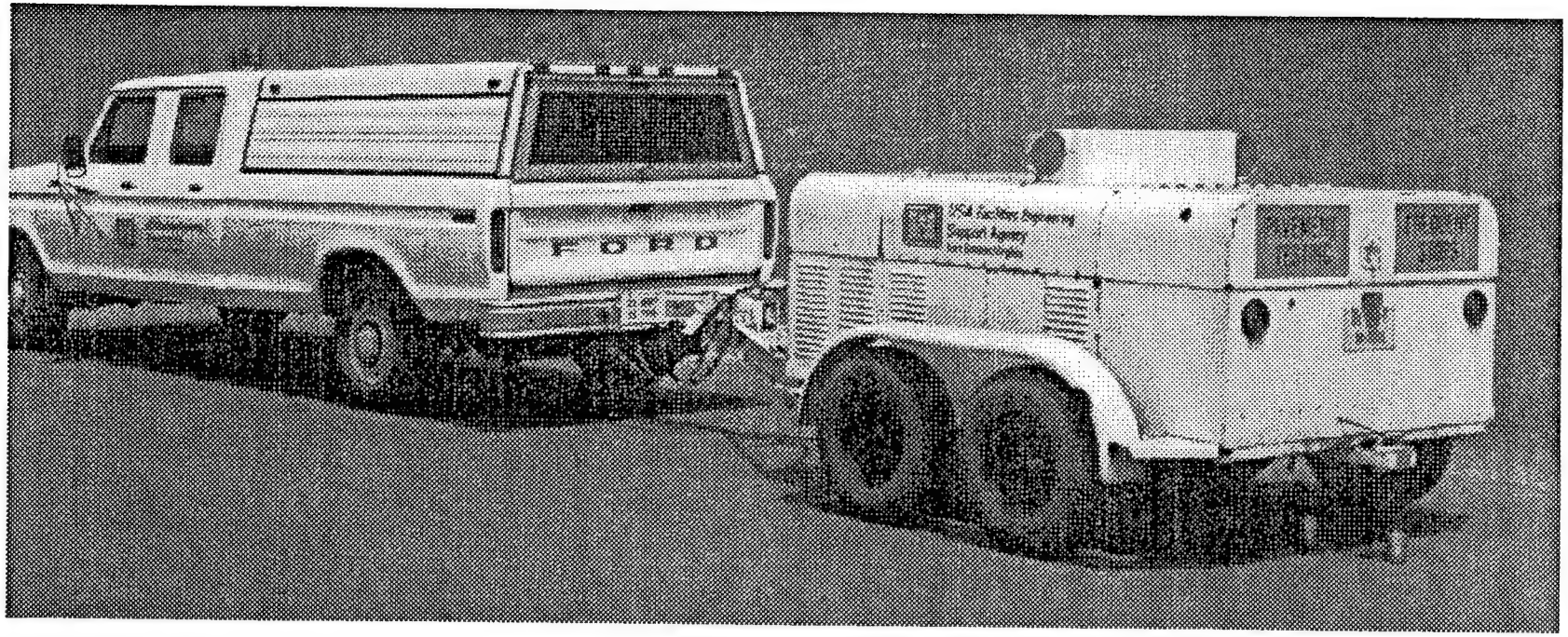


Figure 4. Steady-state dynamic load applied by the Dynaflect



Photograph 2. The Road Rater Model 2008

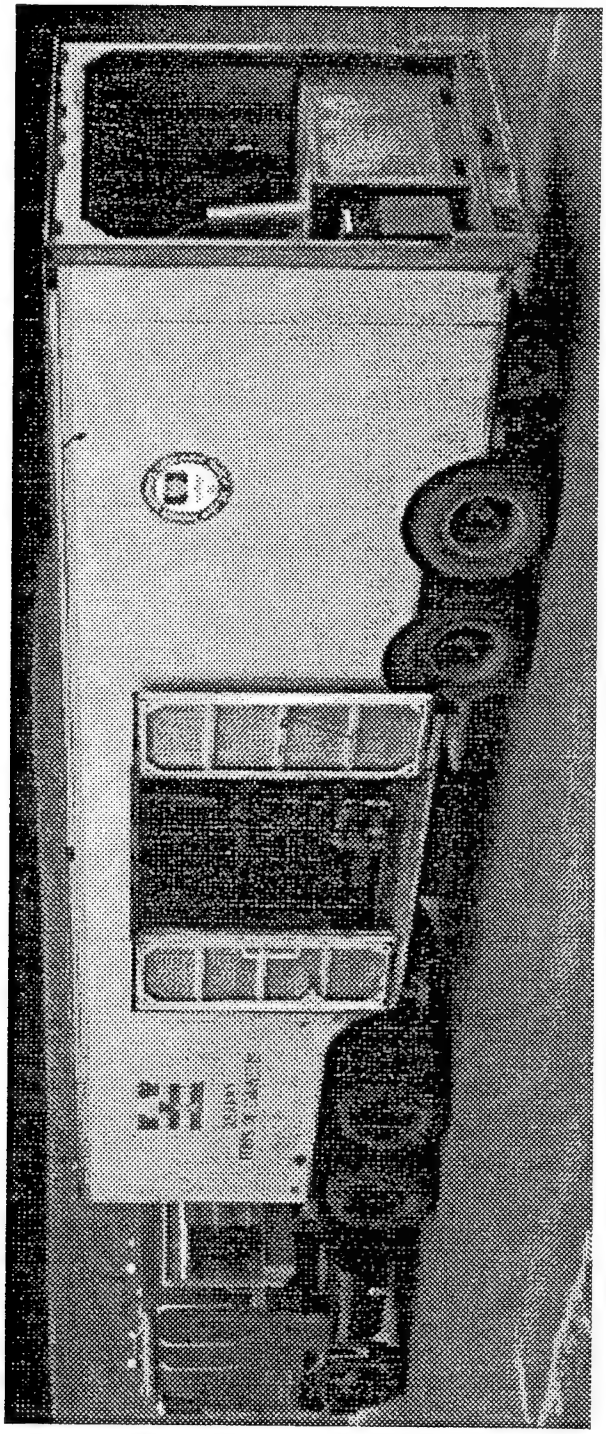
with a peak-to-peak range between 1000 lb and 7000 lb and a frequency between 5 Hz and 70 Hz.

The WES 16-kip Vibrator (Photograph 3) is similar to the Road Rater but has a reaction mass of 16,000 lb and can generate a harmonic load with a peak-to-peak range of 30,000 lb. Unlike the Road Rater and the Dynaflect, it is too heavy to be towed by conventional vehicles. Instead, as the photograph shows, it is mounted in a dedicated tractor-trailer rig along with all of the necessary electronic equipment.

All three steady-state devices are a significant improvement over the static devices; however, they still fall short of replicating the true dynamics of a moving vehicle. The load imposed by a moving vehicle covers a broad range of frequencies rather than a single frequency. Also, the vehicle loads of interest are much higher than can be generated with all but the heaviest of these devices. These drawbacks are addressed by the impulsive (transient dynamic) loading devices described next.

Transient Dynamic NDT Devices

The transient dynamic NDT devices apply an impulse load to the pavement and record the resulting pavement deflection histories at several radial distances from the load (Figure 5). This experimental data is usually summarized by a "deflection basin" that is constructed from the peak deflections at each measurement location (Figure 6). Because the pavement



Photograph 3. The WES 16-kip Vibrator

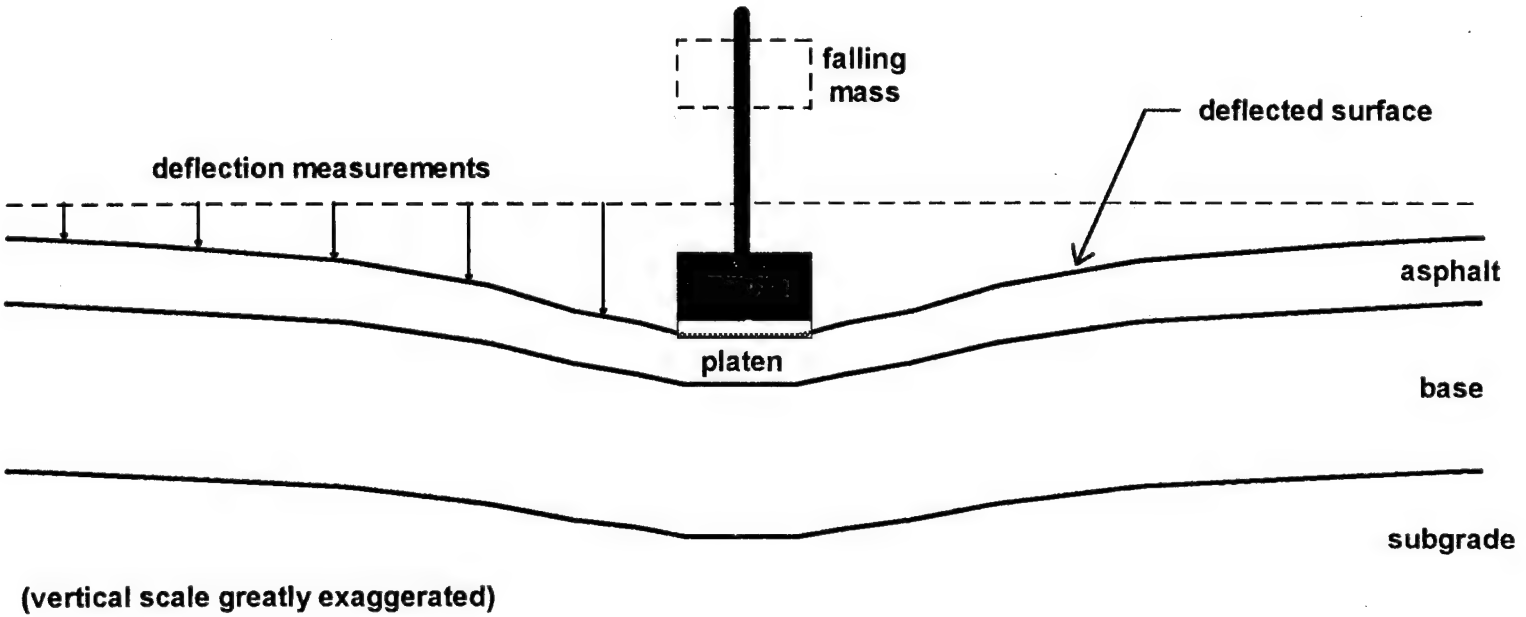


Figure 5. Schematic of a transient dynamic NDT

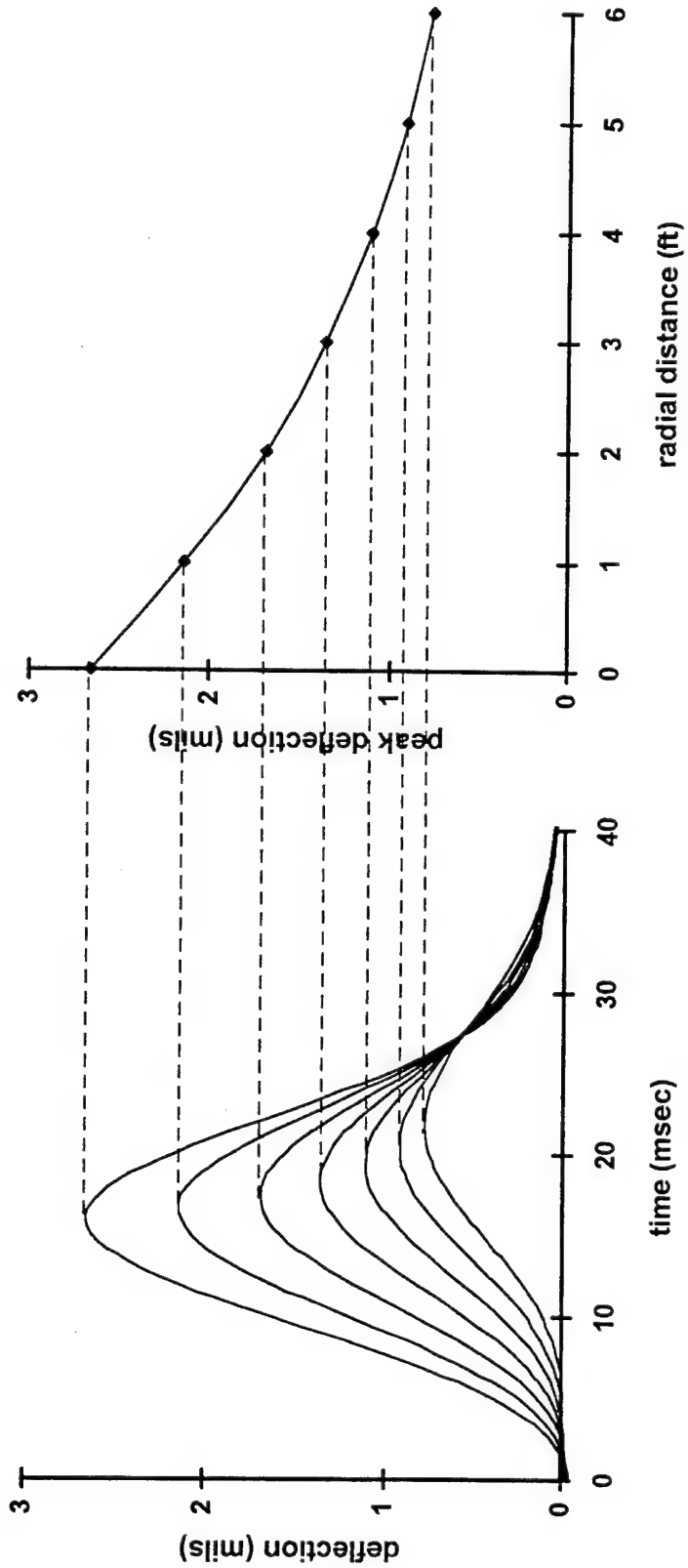


Figure 6. Typical FWD deflection pulses and corresponding deflection basin

deflections at each sensor peak at different times, this deflection basin is more of a presentation tool than a physical entity. The actual deflected shape of the pavement at any point in time is generally much different than the recorded deflection basin (Figure 7).

The Dynatest and KUAB falling weight deflectometers (FWD) exemplify the transient dynamic devices. Both devices generate impulse loads through the rapid deceleration of a falling mass. Each FWD manufacturer uses slightly different means of controlling the deceleration in order to produce a load pulse with the appropriate shape and duration. The goal, of course, is to generate a load pulse similar to that produced by a moving vehicle. A haversine (Figure 8) is sometimes used to represent an idealized loading pulse. That practice will be discussed in more detail in Chapter 5.

The Dynatest FWD (Photograph 4) is available in several models that differ primarily in the peak loads they can produce. The most common, the Model 8000, can produce a load pulse with a peak dynamic force of between 1,500 and 24,000 lb and a duration of 25–30 ms. This is accomplished by dropping a mass (weighing between 110 and 660 lb) from one of four different heights ranging up to 16 in. The mass drops onto a set of rubber buffers that condition the load to produce the desired pulse shape (Figure 9). The load is transmitted from the rubber buffers to the pavement through an 11.8-in-diameter steel plate underlain by a rubber pad. The pad helps distribute the

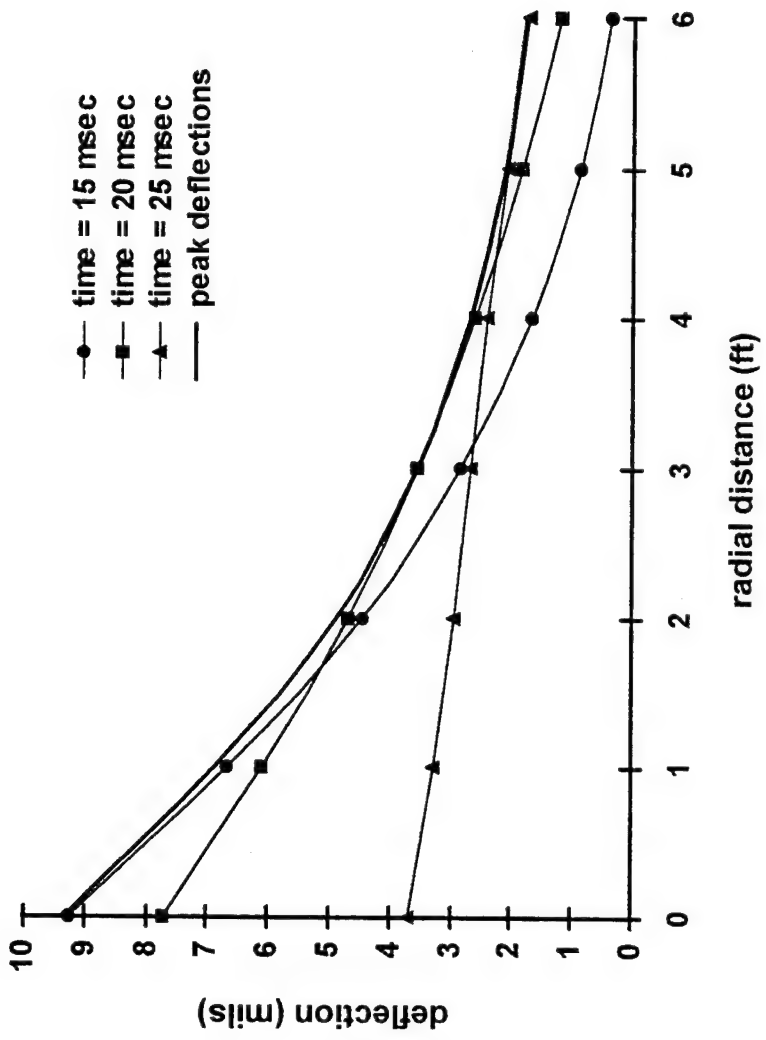


Figure 7. Typical pavement deflections at different times

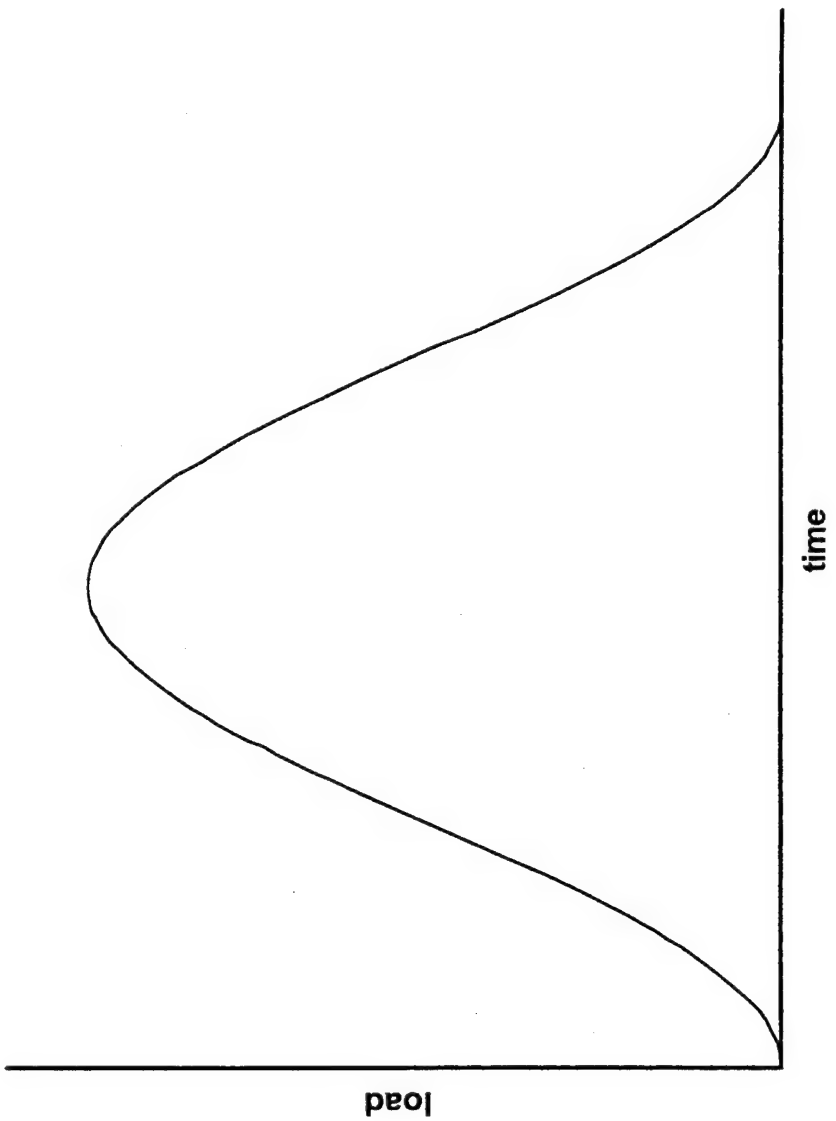
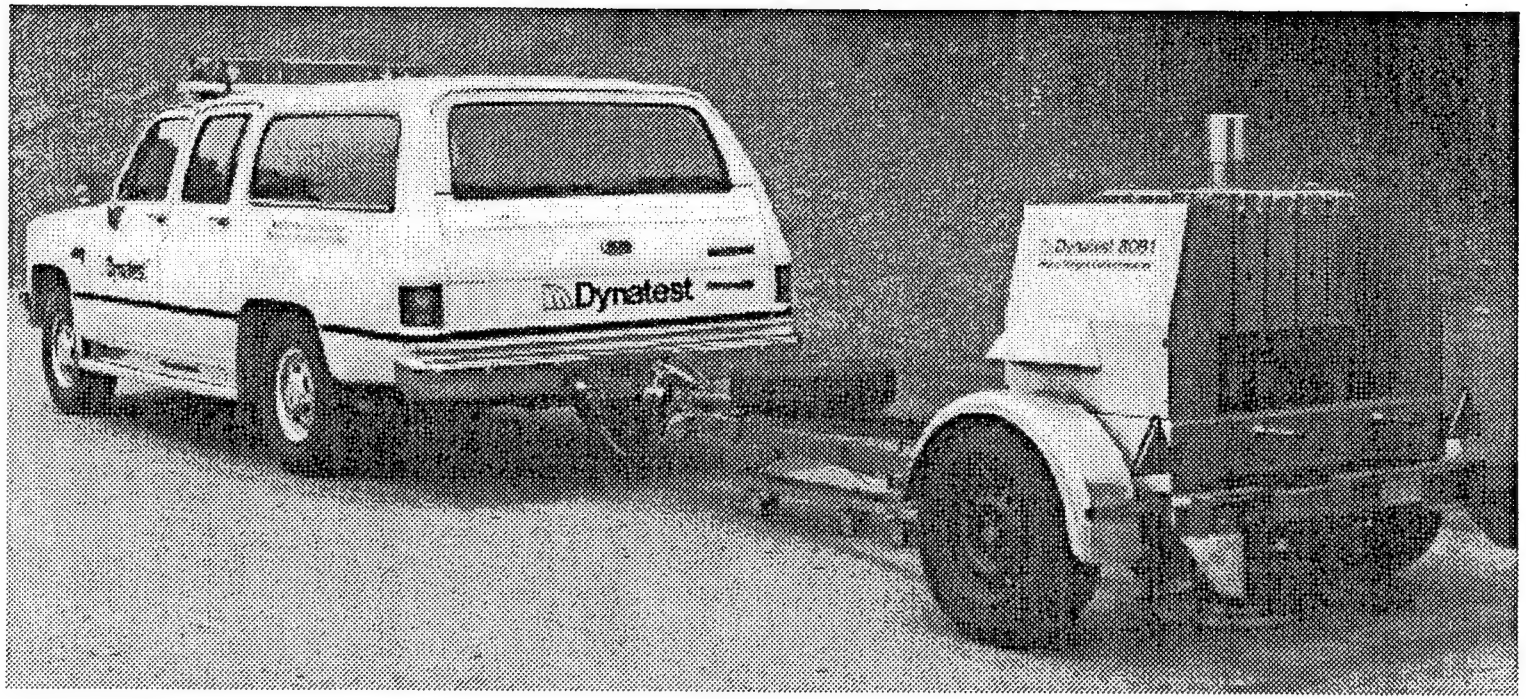


Figure 8. Shape of an idealized FWD loading pulse



Photograph 4. The Dynatest Model 8081 FWD

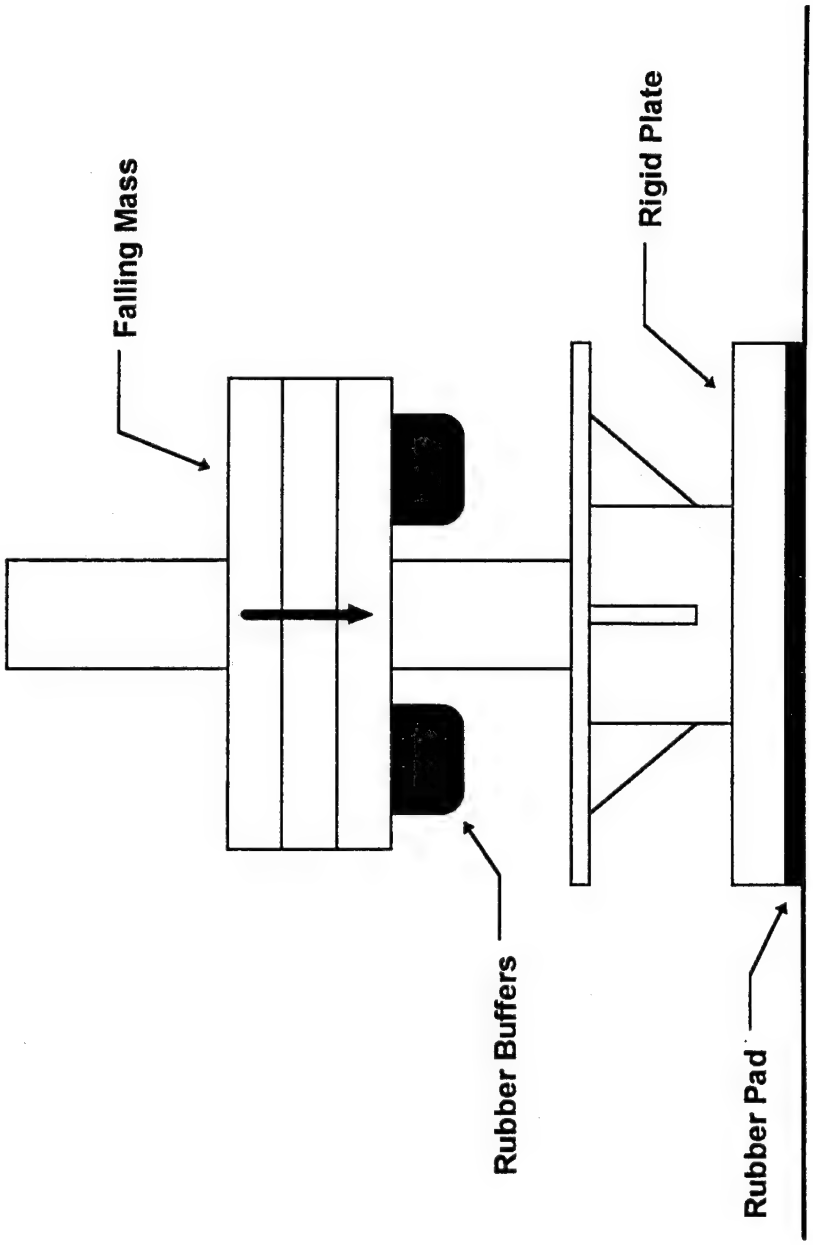


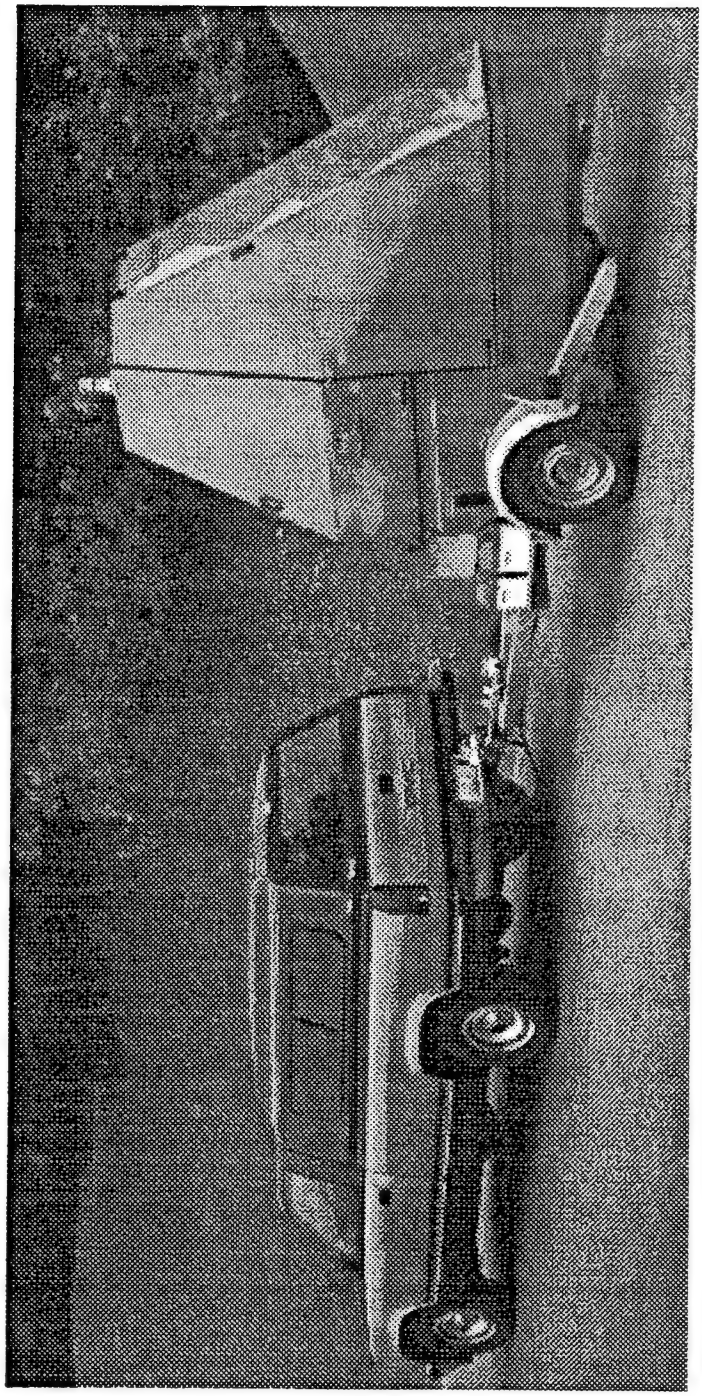
Figure 9. Schematic of the Dynatest FWD

load uniformly to the pavement; its size and shape are meant to approximate the loaded area of a dual rear wheel. The deflections are measured by a series of seven velocity transducers. The first velocity transducer is positioned to measure the pavement velocity at the center of the rigid plate. The others are usually spaced at 1-ft intervals although some agencies, such as the Federal Highway Administration (FHWA), prefer a non-uniform spacing that concentrates more deflection sensors close to the load.[2]

The KUAB FWD (Photograph 5) is also available in several models; they provide operating ranges similar to those of the Dynatest FWD. Unlike the Dynatest FWD, however, the KUAB FWD uses a two-mass system in which one free-falling mass is dropped onto a stationary mass/buffer combination (Figure 10). The two-mass system creates a load pulse with a much longer duration than is obtained with the Dynatest FWD. It also produces a smoother and more-consistent load pulse. The Dynatest FWD can produce a double-peaked load pulse (Figure 11) whose rise time varies depending on the pavement stiffness. The KUAB FWD, on the other hand, produces a much more symmetric, single-peaked load pulse with a fairly consistent rise time and duration of 28 and 56 ms, respectively, on most pavements.

---

[2] According to the National Highway Institute's "Pavement Deflection Analysis" workbook, the sensor locations recommended for use in the Strategic Highway Research Program (SHRP) are 0, 8, 12, 18, 24, 36, and 60 in.



Photograph 5. The KUAB Model 50 FWD

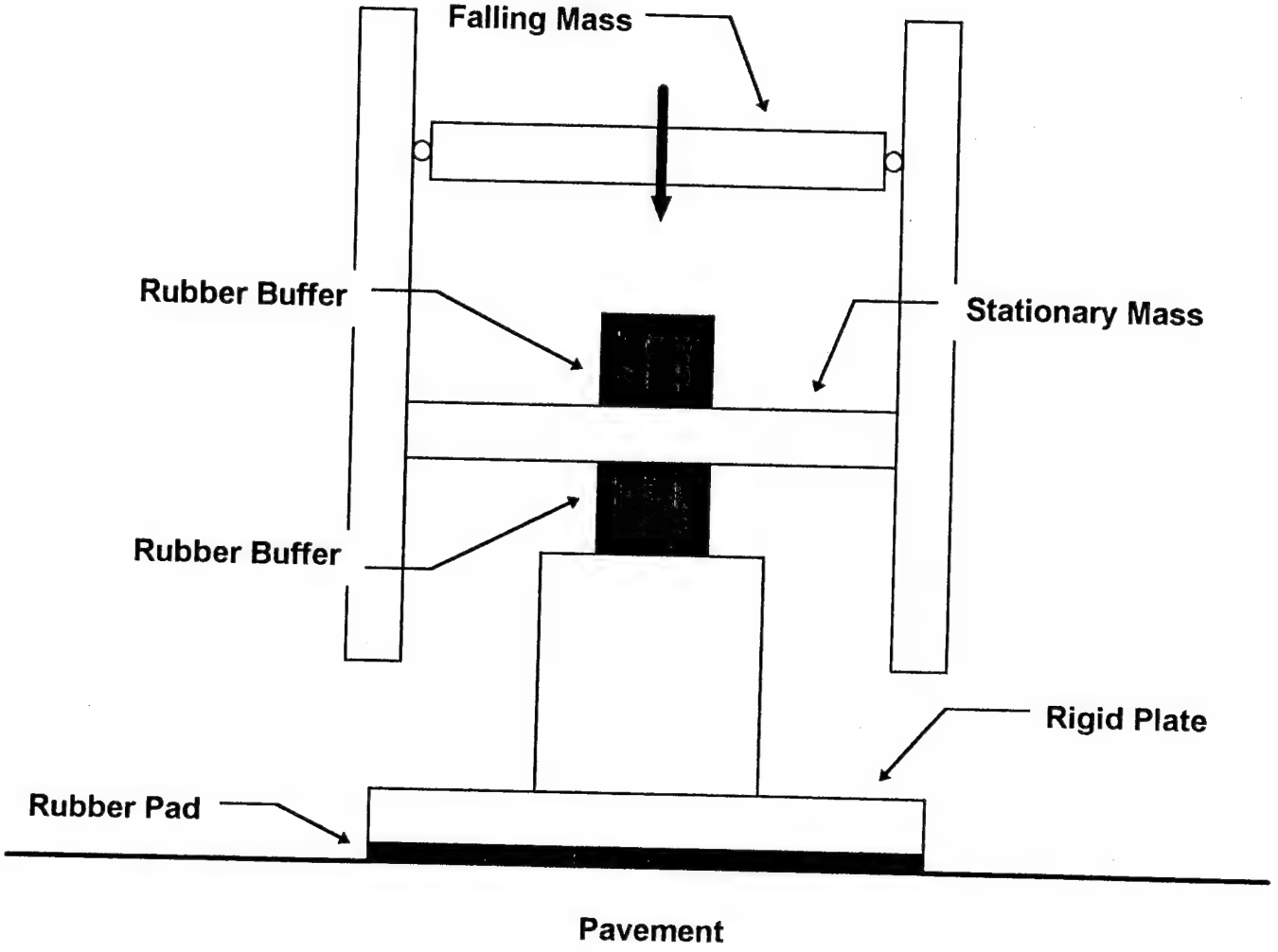


Figure 10. Schematic of the KUAB FWD

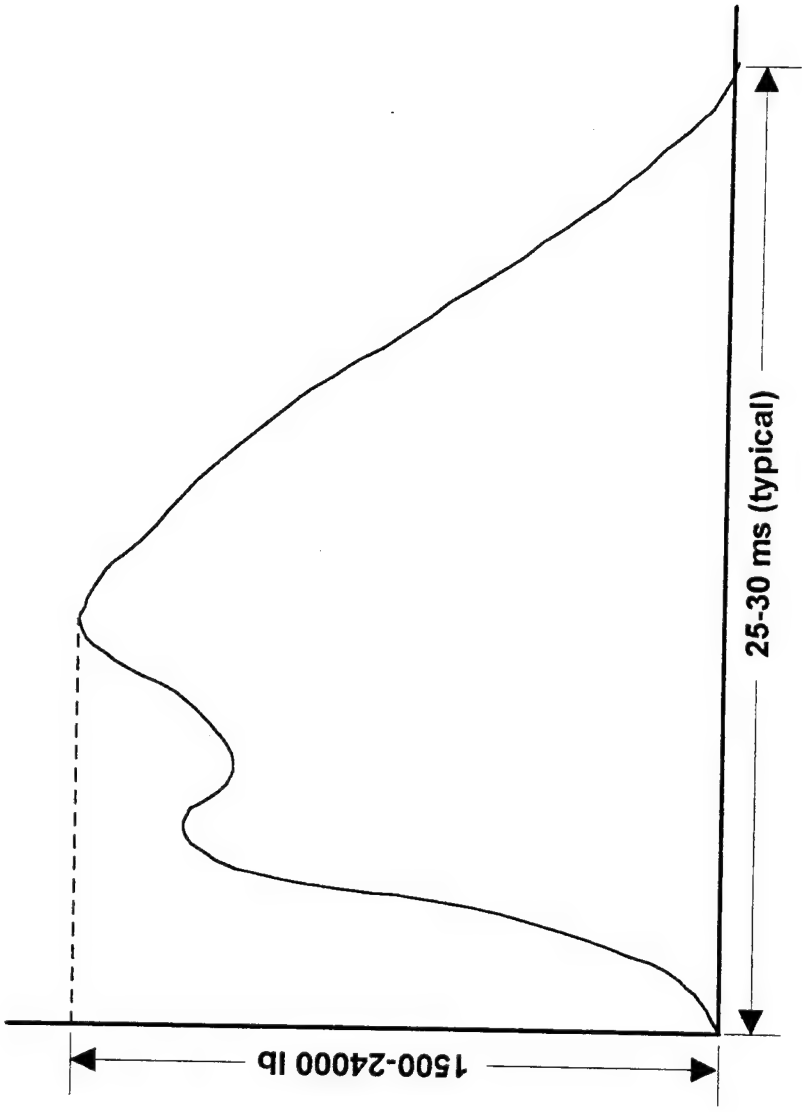


Figure 11. Shape of a typical Dynatest FWD load pulse

One advantage that these devices have over the steady-state devices is that they do not need a static preload. Their force levels are achieved by the rapid, but controlled, deceleration of a relatively light falling mass. As a result, pavement responses to the heavy wheel loads of most interest can be obtained with an extremely light, trailer-mounted device that can be towed by most conventional vehicles. For example, the Dynatest Model 8000 can produce a peak load of 24,000 lb and yet weighs just 2000 lb. Of the steady-state devices, only the WES 16-kip Vibrator can produce a peak load that high and it weighs eight times as much.

Another advantage of the transient dynamic devices is that they come closer than any of the other device types to replicating the force histories and deflections produced by moving vehicles. Bohn, et. al. (1972) showed that the load pulses produced by an FWD are virtually identical in shape and duration to those produced by a truck moving at 25 mph. They also showed that the magnitudes of the resulting deflections were almost identical (Figure 12) despite a significant difference in their duration. These results were confirmed by Hoffman and Thompson (1982), who showed that the duration of the deflection pulse produced by a moving vehicle is a function of both the vehicle speed and the pavement stiffness. They pointed out that the duration of the deflection pulse should be roughly equal to the time needed to traverse a distance equal to the width of the deflection basin. (The pavement surface at a

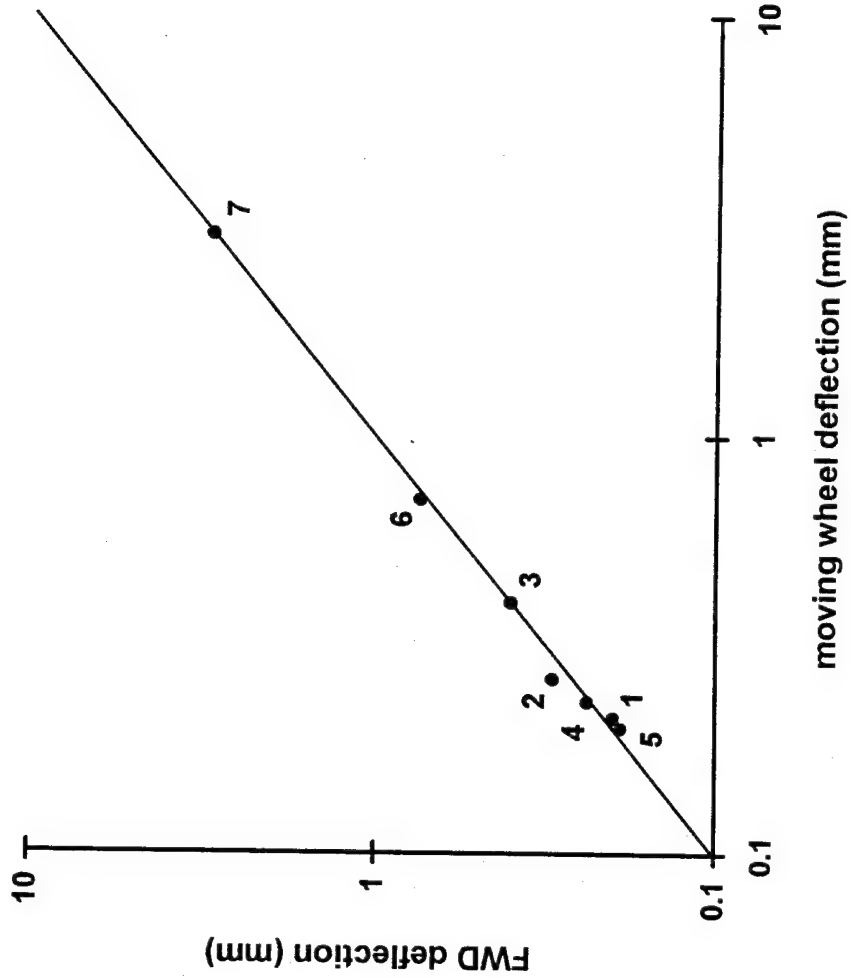


Figure 12. Comparison between FWD and moving wheel deflections (after Bohn, et. al., 1972)

fixed location will start to deflect when the approaching vehicle is one basin radius away and will return to its original elevation (assuming there is no permanent deformation) once the vehicle is one basin radius removed in the opposite direction). Because the width of the deflection basin is inversely related to the stiffness of the pavement system, the duration of the deflection pulse increases with decreasing pavement stiffness. Interestingly, both Bohn, et. al. (1972) and Hoffman and Thompson (1982) found that the magnitude of the deflection pulse is independent of the vehicle speed even though the duration of the deflection pulse is not.

Because the FWD comes closest to replicating the force history and deflection magnitudes of a moving truck, it has gained very wide acceptance. It is currently used by 36 state Departments of Transportation[3]; it has been used by the United States Army since 1980 (Bush, Brown, and Bailey, 1989) and has been adopted for use in the Army Airfield Evaluation Program (Beaucham, 1993); it has been used by the United States Air Force since 1986 (Bush, Brown, and Bailey, 1989) and has been adopted for use by their pavement evaluation teams (Walrond and Christiansen, 1993); and it has been chosen for use in the Strategic Highway Research Program (SHRP) Long Term Pavement Performance (LTPP) study (Richter and Rauhut, 1989).

---

[3] Personal communication with Nick Coetzee of Dynatest, Inc.

Traditional Approaches to FWD Backcalculation

When the FWD (or any other NDT method) is used for the structural evaluation of a pavement system, the goal is to backcalculate the elastic (Young's) moduli of the individual pavement layers from the experimental deflection basins. Unfortunately, there are no closed-form solutions available to accomplish this task. Instead, a mathematical model of the pavement system is constructed and subjected to the appropriate NDT loads in order to determine the surface deflections as a function of the pavement layer properties. This pavement model is then run with a variety of different "trial" layer properties until a set of properties is found which causes the measured deflection basin to be reproduced (Figure 13).

The sophistication of the techniques used to adjust the trial pavement layer properties (ranging from trial-and-error to numerical optimization techniques) often determines the speed with which a solution can be found. The precision of the model (the degree to which similar inputs produce similar answers) is determined in large part by the criteria used to assess convergence. A "loose" convergence criterion provides for a faster but less precise solution. The accuracy of the final solution (the degree to which the answers correspond to reality) is determined in large part by the realism of the pavement system model. There is almost always a trade-off between realism and convergence speed, as well.

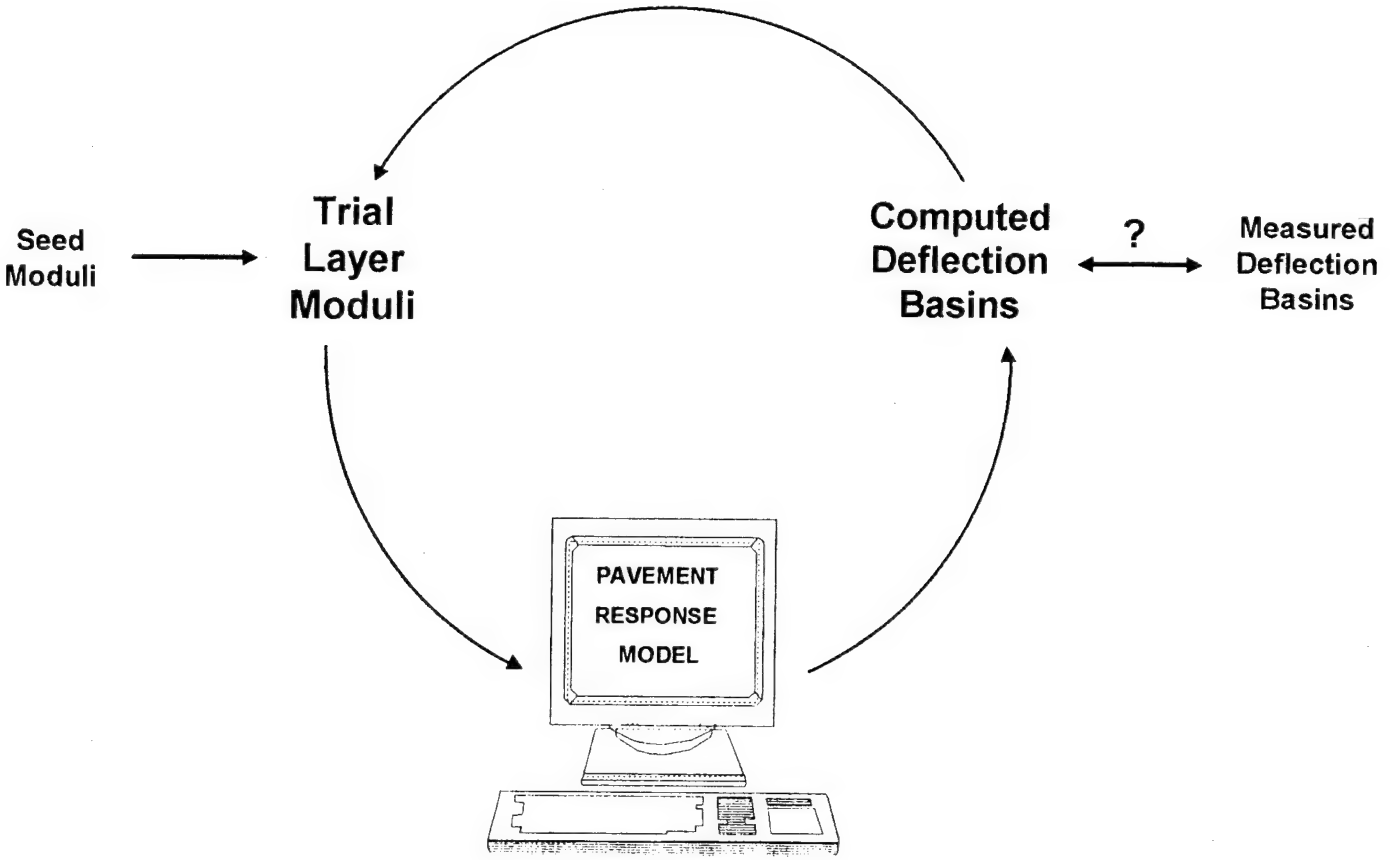


Figure 13. Traditional iterative backcalculation procedure

The first non-empirical backcalculation program for flexible pavements was a convergent trial-and-error solution for two-layered systems developed by Scrivner, Michalak, and Moore (1973). Their solution was based on the theory of stresses and displacements in layered elastic media developed by Burmister (1945a, 1945b, 1945c). It could be used to determine the elastic moduli of the pavement and subgrade given two pavement deflections and the pavement thickness. According to Lytton (1988), the first non-empirical backcalculation procedure for general multi-layer pavement systems was developed by Yih Hou (1977). His approach was to use a gradient-search algorithm to find the set of moduli that minimized the sum squared error between the theoretical and experimental deflection basins.

Most of the backcalculation programs currently in use employ some type of gradient search technique. An alternative approach is to interpolate within a database of theoretical basins covering the anticipated range of pavement layer moduli. These two approaches are discussed next.

Gradient Search Methods

The MODCOMP program developed by Irwin (1991) and the "_DEF" series of programs originated by Bush (1980) typify the gradient-search approach. Each has, at its core, a layered-elastic computer program used to calculate the stresses and strains induced in the pavement system by the applied load. MODCOMP is built around the CHEVRON computer program

(Michelow, 1963). So, too, is CHEVDEF, which was the original _DEF program. As an alternative to using CHEVRON, which assumes that the layer interfaces are fully frictional (i.e., the horizontal displacements on either side of the layer interface are equal), a second _DEF program, BISDEF, was created using the BISAR program (DeJong, Peutz, and Korswagen, 1973). BISAR improves on CHEVRON by allowing the layer interface conditions to be varied smoothly between the two extremes of fully frictional and completely frictionless.

The latest program in the _DEF series is WESDEF (Van Cauwelaert, et. al., 1988), which is based on the layered elastic analysis program WESLEA (Van Cauwelaert, Delaunois, and Beaudoint, 1986). WESLEA improves on BISAR by implementing layer interfaces that obey Coulomb's law. The frictional properties of the layer interfaces can be varied by adjusting the friction coefficient. When the shear stresses at the interface violate Coulomb's law, the interface friction becomes small and slip between the layers is allowed. WESLEA also improves on BISAR by using a combination of closed-form and numerical integration to achieve faster computation speeds without a loss of accuracy. Van Cauwelaert, et. al. (1988) showed WESDEF to be nearly five times faster than BISDEF.

All of the _DEF programs use the same iterative approach to finding the set of pavement layer moduli that minimizes, in an average sense, the errors between the calculated and measured deflections. The _DEF programs begin

by calculating two deflection basins corresponding to a set of user-supplied minimum layer moduli and a set of "seed" moduli. The latter provides a starting point for the solution and the former serves to bound the solution from below. (A set of maximum layer moduli must also be supplied by the user to bound the solution from above.) Using the multi-variate equivalent of linear interpolation (Figure 14), a new set of estimated layer moduli is obtained from these first two deflection solutions. The process is repeated until the average relative error for the seven sensor locations falls within a specified tolerance (e.g., 10 percent). According to Bush and Alexander (1985), convergence is generally obtained within three iterations.

Database Methods

The MODULUS program (Uzan, Lytton, and Germann, 1988) is an example of the database approach. A database of calculated basins is first generated for a prescribed set of pavement layer thicknesses by parametrically varying the pavement layer moduli within specified ranges. The BISAR program was originally used to calculate these theoretical deflection basins. Versions are currently available that use the static, linear-elastic WESLEA program as well as viscoelastic and finite-element programs.

Once the database of theoretical basins has been constructed, MODULUS uses the Hooke-Jeeves pattern searching algorithm (Letto, 1968) to determine the deflection basins in the database that most closely match the

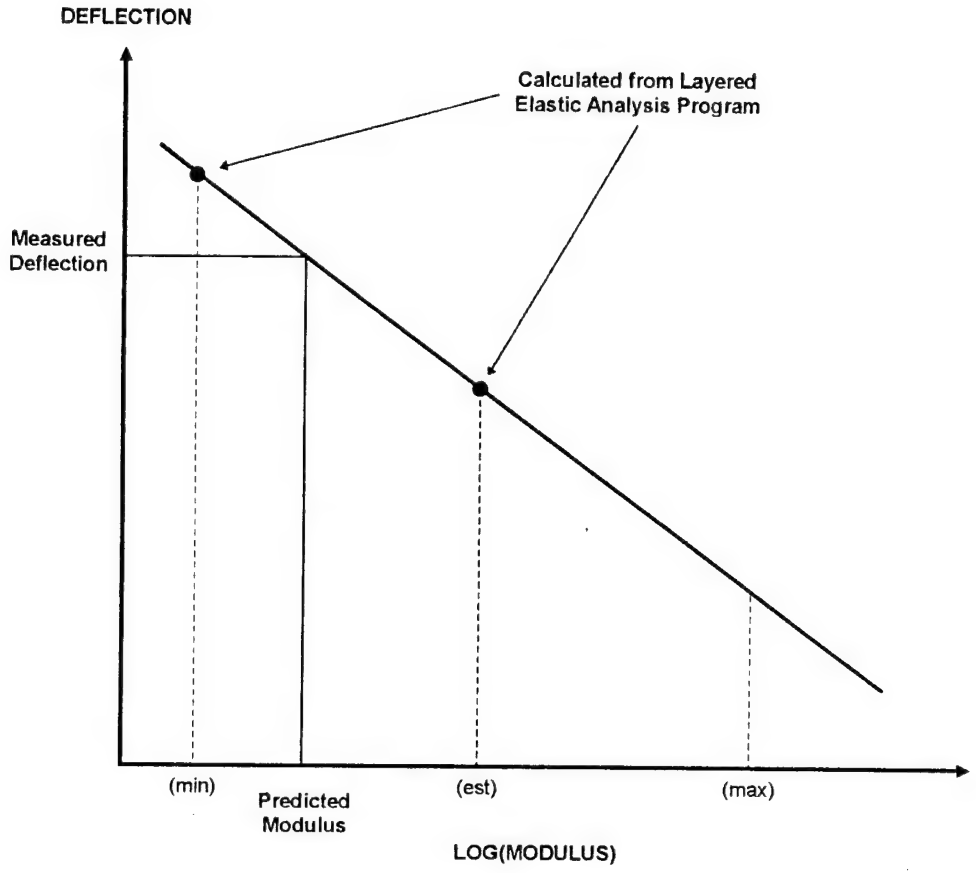


Figure 14. Simplified description of the _DEF iteration scheme
(after Bush and Alexander, 1985)

experimental basin. It then calculates the layer moduli corresponding to the experimental basin using three-point Lagrange interpolation.

MODULUS trades the iterative solution procedure of the gradient-search methods for an iterative database generation procedure. As long as the pavement layer thicknesses used to generate the database do not change, MODULUS can rapidly process FWD test results because each solution is simply calculated by interpolating within the database. If there are only one or two tests in each pavement section, however, MODULUS may actually be less efficient than its gradient-search counterparts (e.g., BISDEF) because new databases must continually be generated.

Drawbacks to Traditional Backcalculation Methods

Traditional backcalculation methods are inherently slow. Part of their inefficiency stems from the iterative nature of the solution process. That alone would not be a problem were it not for the mathematical complexity of the pavement system models. The calculation of the surface deflections requires the numerical integration of transcendental stress functions (Bessel functions) and the solution of systems of simultaneous equations representing the boundary conditions of the individual pavement layers. This computational complexity, when coupled with the need to iterate on the solution, hinders all attempts at real-time backcalculation. It also precludes the use of more realistic (and even more complex) mathematical models of the FWD test.

The FWD test is inherently a *dynamic* test. With few exceptions, however, traditional basin-matching programs employ *static* deflection basin calculations because they are orders of magnitude faster to run[4]. The response of a pavement system to the dynamic loads produced by an FWD can differ significantly from its response to static loads due to resonance in the subgrade and inertial effects. As a result, pavement layer moduli that are backcalculated by matching static theoretical basins to dynamic experimental basins may be significantly in error.

Davies and Mamlouk (1985) were among the first to study this problem. Using a numerical solution for the dynamic response of a layered elastic system to a steady-state load (such as would be applied by the Road Rater) they explored the differences in surface deflections produced by static and dynamic loads for a wide variety of pavement layer properties. Typical of their results are the curves shown in Figure 15. These indicate that a static analysis of pavement response can be in error by 20–30 percent or more. For the pavement system with the thick surface layer, the dynamic deflections are only three-fourths as large as would be predicted using a static analysis. This

---

[4] A recent exception is the FWD-DYN program (Foinquinos, Roesset, and Stokoe; 1993a) which has an option for backcalculating layer moduli using a dynamic analysis. It is, however, very time-consuming to run and requires complete deflection *histories* rather than deflection basins.

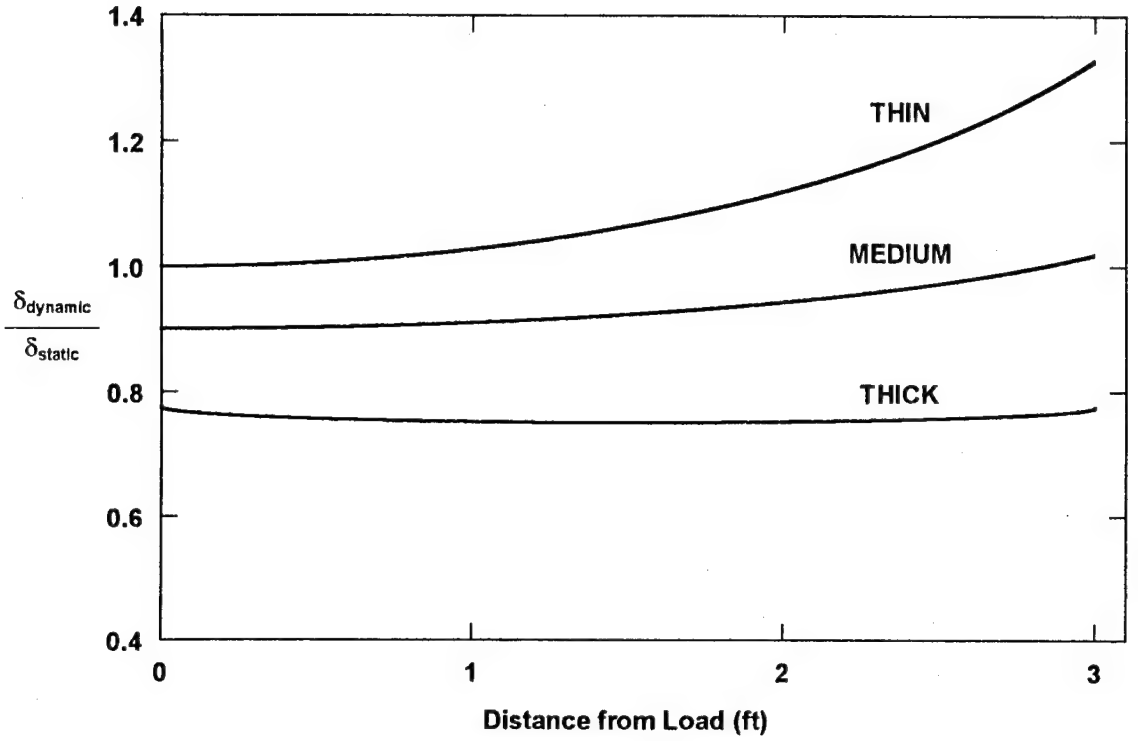


Figure 15. Ratio of dynamic to static deflections as a function of distance from load (after Davies and Mamlouk, 1985)

means that layer moduli backcalculated using a static analysis could be *over*predicted by as much as 30 percent. On the other hand, the dynamic deflections of the pavement system with the thin surface layer are larger than would be predicted using a static analysis. The discrepancy becomes more pronounced as distance from the load increases. At a distance of 3 ft, the dynamic deflections are almost 40 percent larger than the corresponding static deflections. This could result in an *under*prediction on the order of 30 percent.

Sebaaly, Davies, and Mamlouk (1985) used the principles of Fourier superposition to perform a dynamic analysis of pavement response to the transient loads produced by the FWD. Their elastodynamic solution provided deflections very similar to those produced by both a moving truck and the FWD (Figure 16). In a subsequent paper (Sebaaly, Mamlouk, and Davies, 1986), the same authors showed that deflection basins provided by their elastodynamic solution came much closer to matching experimental FWD deflection basins than did deflections obtained from a static analysis (Figure 17).

Roesset and Shao (1985) performed similar dynamic analyses of the steady-state-dynamic Dynaflect and the transient-dynamic FWD in order to investigate the effects of bedrock depth on the measured deflections. As part of their investigation, they computed FWD deflection basins for a series of four pavement systems that varied only in the thickness of their subgrades using an elastodynamic model of the FWD test. A basin-matching program employing

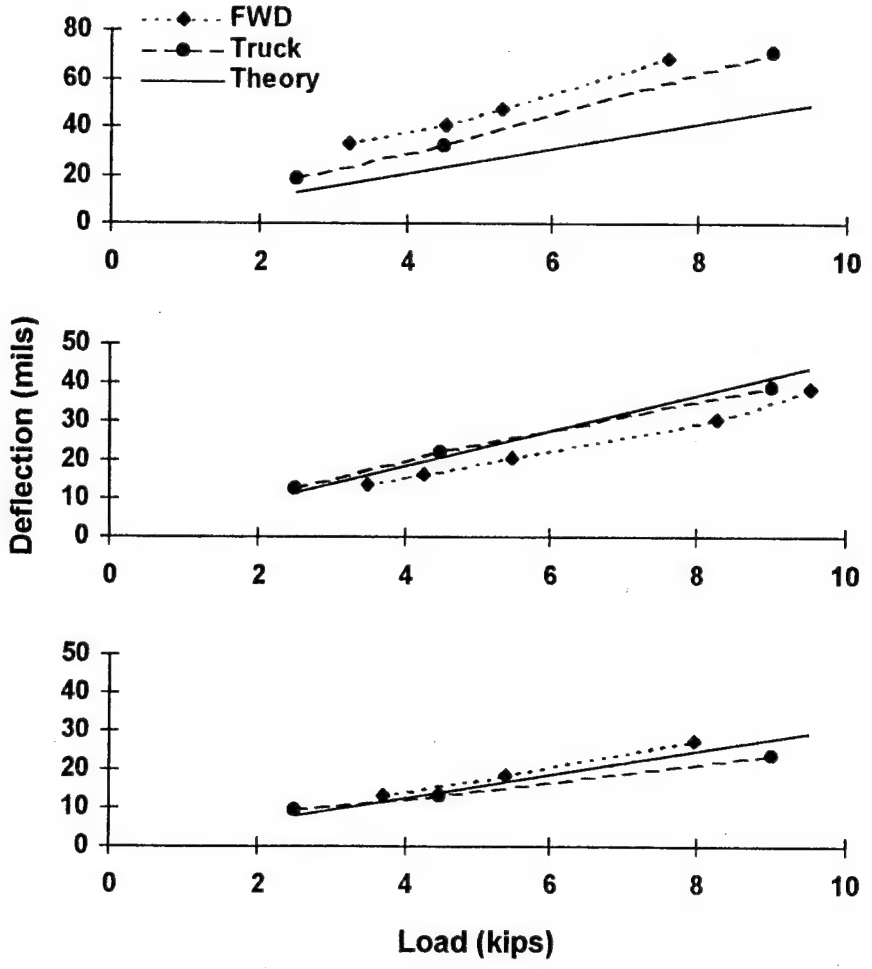


Figure 16. Comparision with theory of deflections produced by a truck and an FWD (after Sebaaly, et. al., 1985)

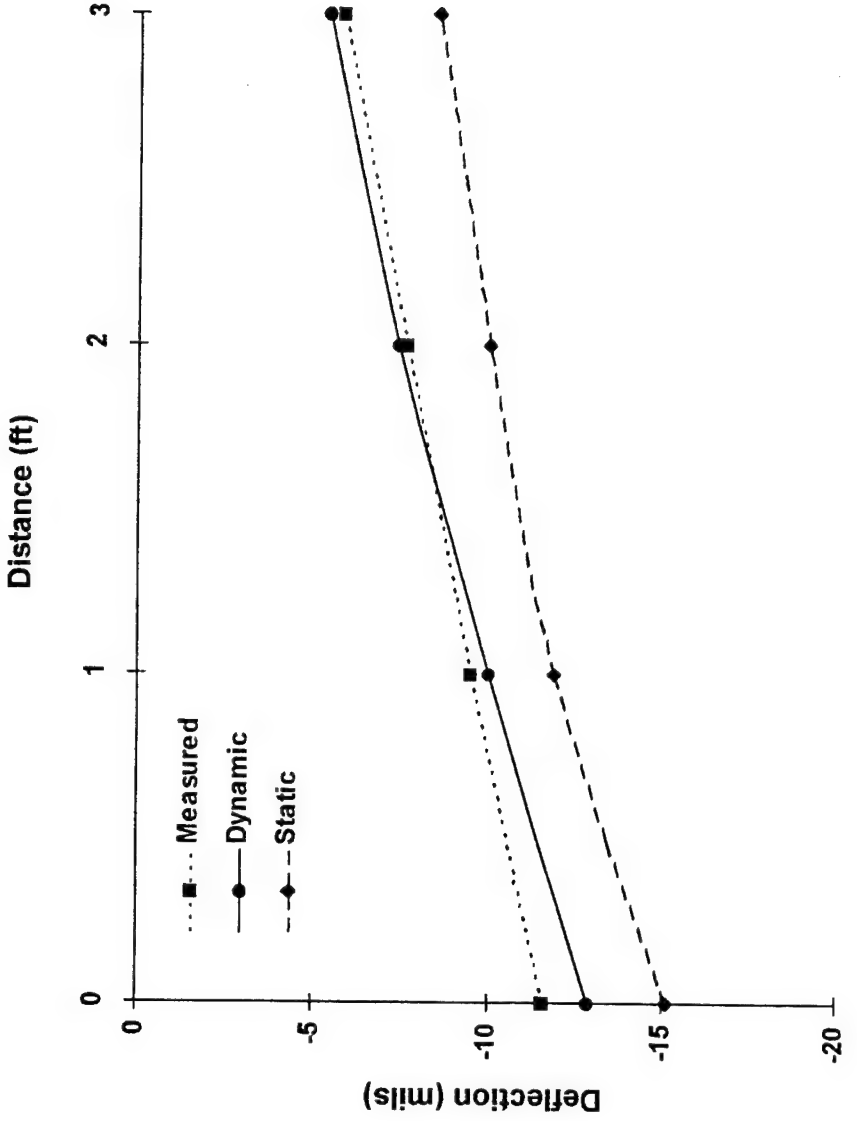


Figure 17. Comparison of measured, static, and dynamic deflections (after Sebaaly, et. al., 1986)

the gradient-descent technique and a *static* model of the FWD test was then used to backcalculate the moduli. The seed moduli used to initiate the iterative solution procedure were the correct moduli, so any errors had to be the result of the program's inability to match the dynamic deflection basins using a static analysis. Their results showed that the errors that accrue from using a static analysis can be quite significant.

Chang, et. al. (1992) also investigated the effects of subgrade thickness on surface deflections. They showed that shallow bedrock invariably produces dynamic amplification (i.e., the peak dynamic deflection is greater than the static deflection). The use of static analyses for those pavement systems can lead to significant underestimation of the subgrade moduli and overestimation of the asphalt and base layer moduli. They also showed that, under the proper circumstances, deep bedrock can produce the opposite effect (i.e., dynamic attenuation). In that case, the subgrade moduli would be overestimated at the expense of underestimating the asphalt and base moduli.

Using an approach similar to that of Roesset and Shao, Chang, et. al. used the basin-matching program MODULUS (which exemplifies the database approach) to backcalculate moduli from dynamic deflection basins computed for three flexible pavement profiles and one rigid pavement profile. Several deflection basins, corresponding to different depths to bedrock, were produced for each pavement profile. Some of their results are shown in Table 1.

Table 1. Ratios of backcalculated moduli to true moduli
(after Chang, et. al., 1992)

Site	Depth, ft	Surface	Base	Subgrade
FM 195	5	1.85	2.16	0.50
	80	11.71	0.88	1.06
FM 137	7.5	1.61	1.66	0.63
	80	1.02	1.19	1.13
Route 1	15	0.97	2.07	0.79
	80	0.99	1.33	0.97
I-10	10	1.25	-	0.67
	80	1.08	-	1.44

Foinquinos, Roesset, and Stokoe (1993b) also investigated the relationship between FWD surface deflections and bedrock depth for both flexible and rigid pavements. They showed that depth to bedrock has significantly less effect on the peak dynamic deflections than on the static deflections (Figure 18). Because depth to bedrock is usually estimated rather than measured, any errors in the assumed depth will be magnified by the static analysis. A dynamic analysis, on the other hand, would be insensitive to bedrock depth for all but the shallowest of depths.

These studies and others like them show the importance of properly modeling the dynamics of the FWD test. Unfortunately, a dynamic analysis of pavement response is far more time-consuming than a static analysis. All dynamic analysis programs use Fourier synthesis to combine steady-state solutions at dozens of different frequencies contained in the transient loading pulse. This alone makes them at least an order of magnitude slower than comparable static programs. As a result, it is extremely impractical to include a dynamic analysis in conventional backcalculation programs.

A New Approach to FWD Backcalculation

This study presents a fundamentally different approach to FWD backcalculation using artificial neural networks. Artificial neural networks are "[computational] systems that are deliberately constructed to make use of some of the organizational principles that are felt to be used in the human brain"

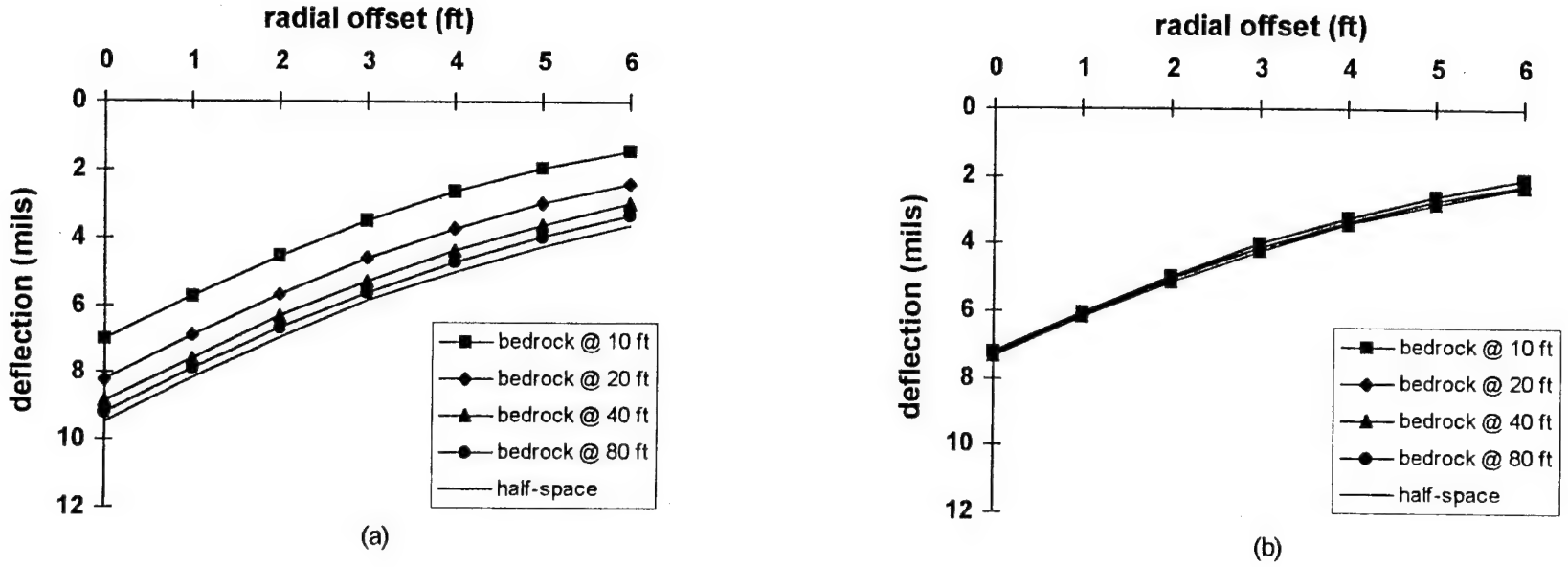


Figure 18. Static (a) and dynamic (b) deflection basins for various depths to bedrock (after Foinquinos, Roesset, and Stokoe, 1993)

(Anderson, 1988). More to the point, artificial neural networks of the type used in this study are universal functional approximators that can "learn" a functional mapping through repeated exposure to examples of that mapping. In fact, Hornik, Stinchcomb, and White (1989, 1990) proved that multi-layer, feed-forward artificial neural networks can be used to approximate *any* continuous function (and certain discontinuous but piecewise-differentiable functions) to any desired degree of accuracy provided that the network is sufficiently large.

In the context of FWD backcalculation, an artificial neural network can be "taught" to map deflection basins back onto their corresponding pavement layer moduli. One way to train such a network would be to use experimentally-determined deflection basins along with independently-measured pavement layer thicknesses and moduli. As mentioned in Chapter 1, it is often difficult to obtain representative, undisturbed samples with which to make a laboratory determination of the pavement moduli. Furthermore, because laboratory testing is expensive, there is an insufficient quantity of experimental data covering a broad-enough range of pavement layer moduli and pavement layer thicknesses to successfully train a neural network.

Instead, synthetic deflection basins calculated using programs such as WESLEA can be substituted for the experimental deflection basins. This allows precise control of the pavement layer properties used to train the network. Furthermore, to the extent that mechanistic pavement analysis programs such

as WESLEA are used to design pavement systems and determine overlay requirements, it has been argued that maintaining consistency between the backcalculation program and the analysis program is preferable to having laboratory-measured moduli (Houston, Mamlouk, and Perera, 1992).

The basic neural network training procedure developed for this study can be viewed as a closed loop (Figure 19). A mathematical model is used to calculate a synthetic deflection basin for a presumed set of pavement layer properties. The artificial neural network is then taught to perform the inverse operation of mapping the synthetic deflection basin back onto the presumed set of properties. At first, the neural network produces a random mapping; however, by repeating the training process many times for many different pavement profiles, the neural network will eventually learn the appropriate inversion function.

Advantages of Artificial Neural Networks

One of the most important advantages to using artificial neural networks is their speed. The mathematical simplicity of neural networks (which will be covered in more detail in the next chapter) makes them computationally efficient. Furthermore, since the computational speed of the trained neural network is independent of the mathematical complexity of the algorithms used to develop the training examples, artificial neural networks can not only perform backcalculation in real-time, they can learn a backcalculation function that is

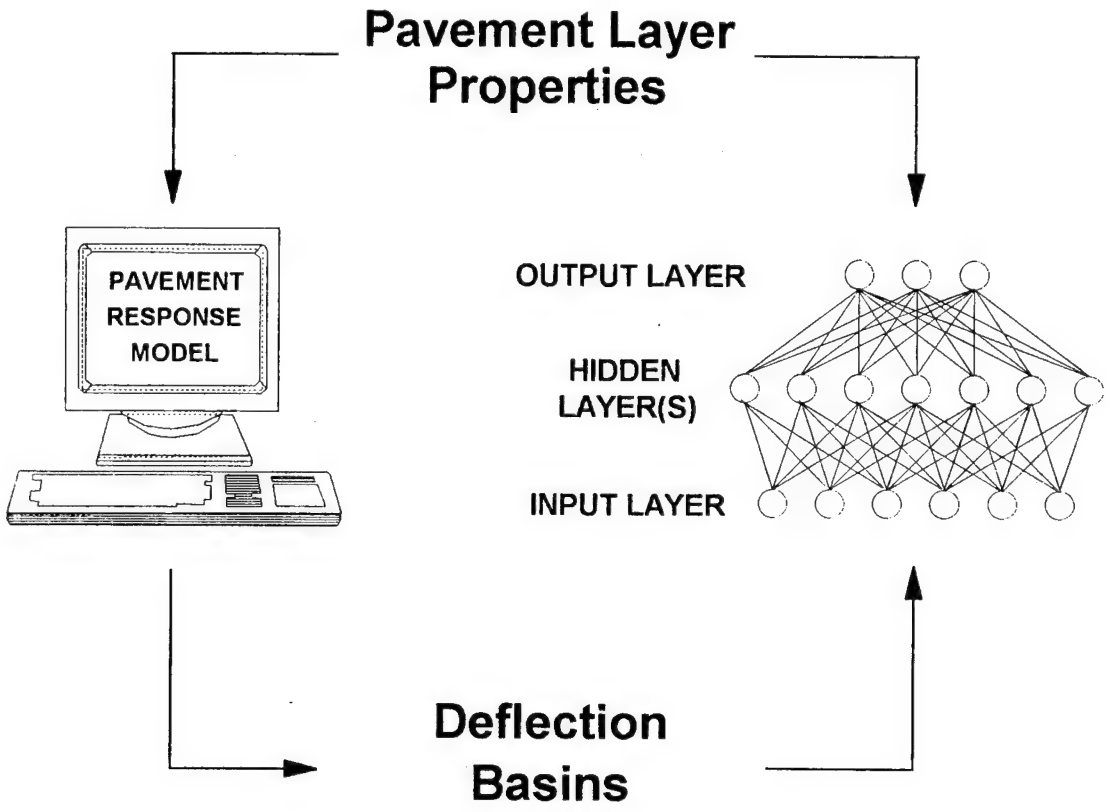


Figure 19. Basic neural network training procedure

based on much more realistic (albeit, more complex) models of pavement response than are used in traditional basin-matching programs. This significantly improves their accuracy without the usual trade-off of decreased computational speed.

Militarily, this is important because both time and accuracy are critical in determining the structural integrity of roads and airfields in a contingency operation. Accuracy is greatly improved by using a dynamic analysis instead of a static analysis. Prior to this research, however, there was always a trade-off between the speed and the accuracy of the analysis. Artificial neural networks offer the possibility of both increased accuracy and increased speed when compared with traditional approaches to backcalculation.

This research also has broad applicability in the civilian sector. State and Federal departments of transportation have thousands of miles of highway that must be periodically evaluated to determine their adequacy and to develop remedial measures if they are found to be inadequate. The frequency of testing is limited by the time needed to perform the test itself and the time needed to evaluate the test results. Real-time backcalculation using artificial neural networks eliminates one of those bottlenecks directly. It also opens up the possibility of developing nondestructive testing techniques that both measure and evaluate pavement properties in real time. Real-time NDT would not only allow more pavement to be tested more often, it would also reduce the costs of

performing the test. It would no longer be necessary to close a traffic lane and divert traffic to perform a test. This would alleviate both the direct costs of traffic control and the indirect costs of commuting delays. Furthermore, the work crews performing the tests would be at much less risk if they did not have to work outside their vehicle in the middle of a traffic lane. The eventual goal of real-time NDT would be to perform the entire test at a reasonable travel speed from the driver's seat.

Summary

This chapter has presented a synopsis of the three primary types of NDT device—static, steady-state dynamic, and transient dynamic—and described the operating characteristics of several devices from each class. The FWD, which is typical of the transient dynamic devices, was chosen for this study because it is widely used by both military and civilian Federal agencies as well as a majority of state Departments of Transportation.

This chapter has also presented a brief description of traditional methods for backcalculating pavement layer moduli from FWD results. Most of those methods entail matching theoretical deflection basins to the experimental basins. The primary disadvantage of these methods is that they are computationally intensive and thus inherently slow. This has generally precluded the use of more-realistic (and more-complex) dynamic response

models. This inability to account for the dynamics of the problem seriously affects the accuracy of their solutions.

A fundamentally new approach to backcalculation based on artificial neural networks has been introduced. Artificial neural networks are universal functional approximators that learn by example. In this study, examples of FWD inversion will be developed by using static and dynamic pavement response models to create synthetic deflection basins covering a wide range of pavement layer properties. Even though it will take considerably longer to develop the training examples with the dynamic model than the static model, the trained neural networks should be able to provide solutions in real time regardless of the pavement response model used to train them.

Prior to this research, there was always a trade-off between the speed and the fidelity of the analysis. Neural networks can circumvent this trade-off between computational speed and computational accuracy because all of the mathematically-complex computations (i.e., the development of the neural network training examples and the training of the network itself) can be performed up front on high-speed computers (in this case, the Cray Y-MP supercomputer at the U.S. Army Engineer Waterways Experiment Station in Vicksburg, MS). Once trained, the neural network represents a simple functional mapping that can be implemented in real time on any personal computer.

CHAPTER 3

MULTI-LAYER, FEED-FORWARD NEURAL NETWORKS

This chapter will provide an historical and mathematical background for artificial neural networks. An exhaustive discussion of the many different types of artificial neural networks and their underlying principles and mathematics is beyond the scope of this work. Instead, this chapter will present a concise chronology of the developments that have led to the multi-layer, feed-forward, artificial neural networks used in this study. In the process, much of the mathematics behind these versatile computational tools will be introduced.

Historical Perspective

The field of neurocomputing has grown out of the scientific desire to understand and explain the workings of the human brain. Initially, neurocomputing was entirely the realm of neurobiologists and psychologists. In fact, most of the seminal writings in the field appeared in journals such as the *Bulletin of Mathematical Biophysics* and the *Psychological Review*. As work in the field progressed, artificial neural networks began to be studied as much for their mathematical, statistical, and computational properties as for the insight

they provided into the workings of the human mind. As engineers and physicists got involved in the study of artificial neural networks, increasingly less emphasis was placed on constructing biologically-plausible neural networks and increasingly more emphasis was placed on applying neural network technology to the solution of complex or otherwise intractable problems. The explosion of artificial neural network applications in the past decade would almost certainly not have occurred had the constraint of biological feasibility not been relaxed.

A Biological Neuron

To understand the early work in artificial neural networks, it is necessary to understand the basic operation of biological neurons. The mammalian nervous system has many different types of neurons. Because early work in neurocomputing was concerned almost exclusively with understanding the behavior of neurons in the cerebral cortex, cortical neurons are the biological prototype for artificial neurons.

A biological neuron (Figure 20) consists of four primary components: a soma (or cell body) that contains the cell nucleus; a single axon that branches out at its end to send signals to other neurons; a tree-like network of dendrites that receives signals from other neurons; and the synapses, or synaptic junctions, where the communication between neurons takes place.

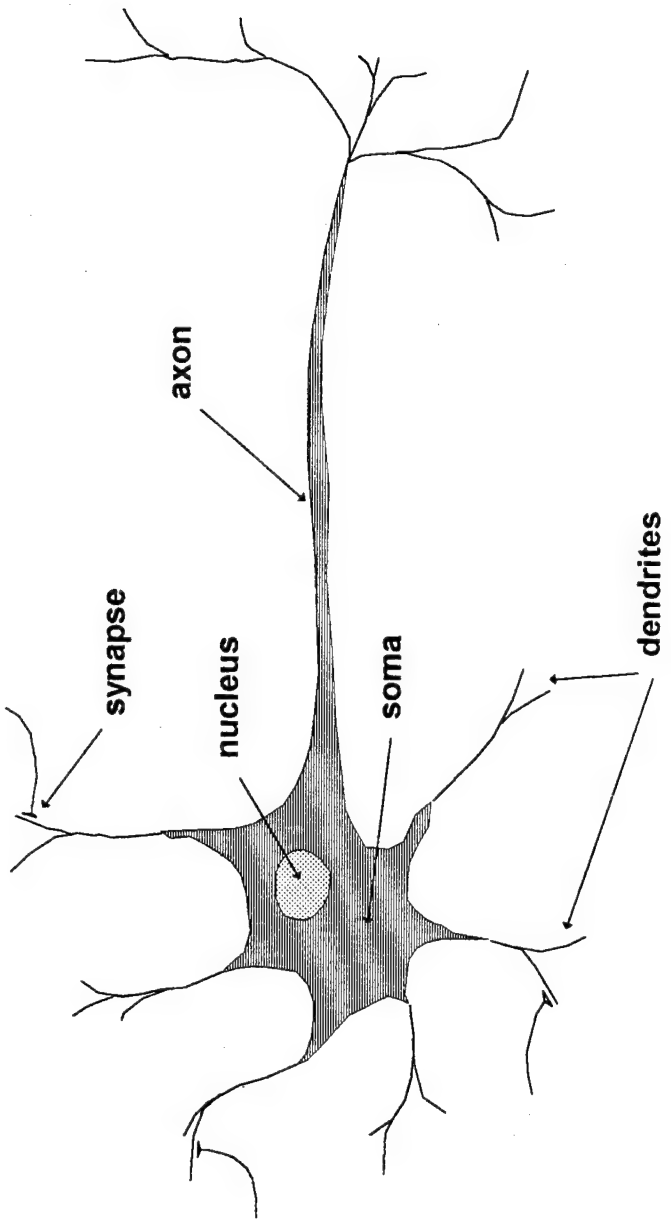


Figure 20. Schematic drawing of a biological neuron

A neuron is continually receiving inputs from other neurons. These inputs arrive in the form of chemical transmitters passed across the synaptic junctions. The chemical transmitters cause either an increase or a decrease in the electrical potential within the cell body. When the electrical potential reaches a certain threshold value, the cell "fires" and sends a signal down its axon to be transmitted to its neighbors.

The McCulloch-Pitts Neuron

McCulloch and Pitts (1943) were the first researchers to define a working model of a neuron. Their seminal work attempted to explain neural processes purely in terms of propositional logic. To that end, they developed a "logical calculus" to explore the behavior of networks of logic elements (i.e., neurons). Their calculus was predicated upon a model of the neuron (commonly referred to now as the "McCulloch-Pitts neuron") that operates according to the following assumptions:

1. The activity of the neuron is a binary process—either the neuron fires or it doesn't fire
2. A fixed number of synapses must be excited within a specific period of time in order for the neuron to become excited
3. There are no significant processing delays other than at the synapses

4. An active inhibitory synapse will completely prevent excitation of the neuron regardless of the level of activity on the excitatory synapses
5. The structure of the network is invariant over time (i.e., new connections do not form and old connections are not broken)

In short, the McCulloch-Pitts neuron is a binary (true/false) threshold unit that fires if the sum of its inputs exceeds a certain threshold (Figure 21). Despite the apparent simplicity of this model, McCulloch and Pitts showed that *any* finite logical expression could be encoded by a network of these binary logic units. Thus, McCulloch and Pitts were among the first to show the power of networks and of parallelism in general.

The Perceptron

The original McCulloch-Pitts neuron was based purely on logic. Their model assumed an equal weighting among the excitatory synapses (which could only be "true" or "false") and allowed a single inhibitory synapse to completely suppress activity. A more general model, based in mathematics rather than logic, was proposed by Rosenblatt (1958), who developed a theory for an intelligent "connectionist system" that he called a "perceptron".

Rosenblatt's work was influenced greatly by Hebb (1949) who theorized an adaptive learning mechanism in which connections between neurons were strengthened if those connections participated in exciting the neurons to fire.

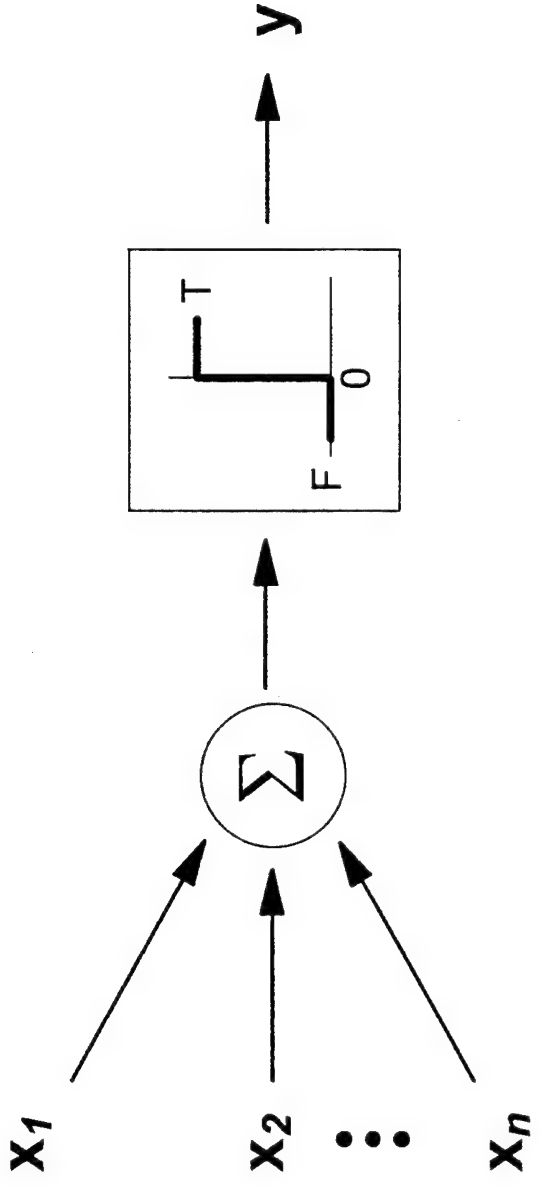


Figure 21. Schematic drawing of a McCulloch-Pitts neuron

Hebb also postulated that information itself, and not just the task of information processing, was distributed throughout the neural system. This lead Rosenblatt to conclude that the information was actually stored in the connections between neurons and that new information was assimilated by modifying those connections. Accordingly, Rosenblatt developed a neuron (which is also rather ambiguously called a "perceptron") that had adjustable weights assigned to each of its synapses and that "learned" (i.e., encoded information) through modification of those weights. In Rosenblatt's perceptron neurons, the magnitude of each weight determines the strength of the connection and its sign determines whether it is excitatory or inhibitory. As in the McCulloch-Pitts neuron, the perceptron fires when the weighted sum of its inputs exceeds a certain threshold (Figure 22).

Mathematically, the perceptron can be modeled as

$$y_j = \Theta(s_j) \tag{1}$$

where s_j is an affine combination of the inputs and the weights:

$$s_j = \sum_i w_{ij} x_i - \beta_j \tag{2}$$

and $\Theta(\cdot)$ is the Heaviside (step) function:

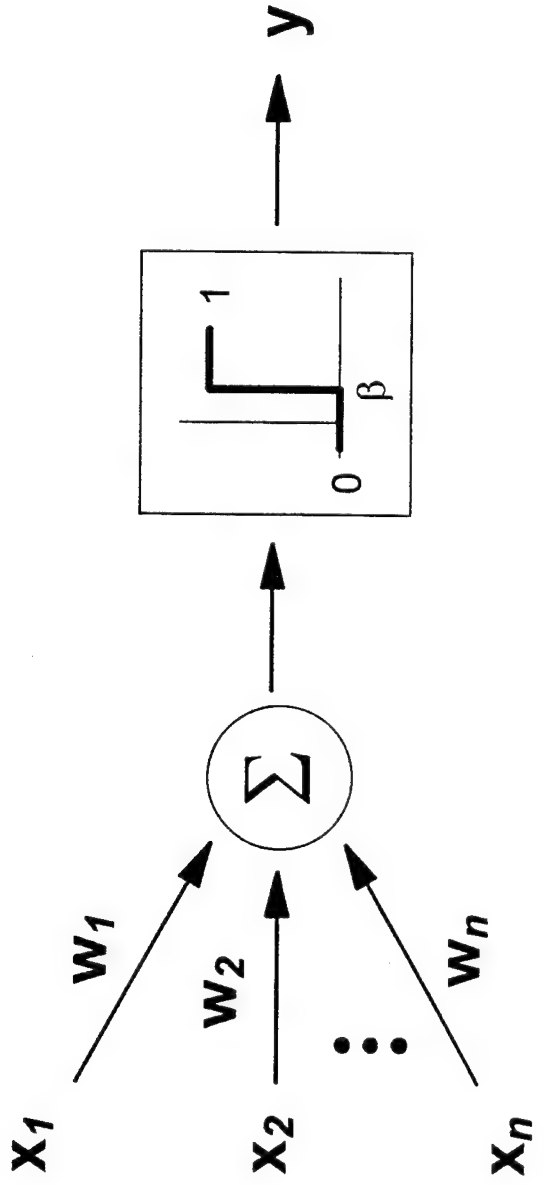


Figure 22. Schematic drawing of a perceptron

$$\Theta(\xi) = \begin{cases} 1 & \text{if } \xi \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Here, the output y_j of neuron j is either 1 (the neuron fires) or 0 (the neuron doesn't fire) depending on the binary inputs x_i received from all of the other neurons in the network. The weight w_{ij} represents the strength of the synapse connecting neuron i to neuron j and the bias β_j represents the threshold that must be reached for the neuron to fire.

The Perceptron Learning Law

Rosenblatt went on to develop a mechanism for training a perceptron in which the perceptron modifies its own synaptic weights in order to produce a desired output. This so-called "perceptron learning law" (Rosenblatt, 1962) is used in conjunction with a process called "supervised learning" to teach a perceptron a pattern classification task. Training involves repeatedly showing the perceptron examples of the patterns to be classified, along with the correct classification of each, and allowing it to adjust its weights so as to produce the desired response to each input. For each exemplar shown to the network, the weights are modified according to the following formula:

$$w_{ij}^{new} = w_{ij}^{old} + (y_j - t_j)x_i \quad (4)$$

where y_j is the realized output and t_j is the target output. Here, the x_i are the elements of a feature vector **x** that describes the pattern that is to be classified. Note that if the realized output (i.e., classification) is correct, the output error $(y_j - t_j)$ will be zero and the weight will not be changed. If the classification is incorrect, the output error will either be +1 or −1 and each connection weight will either be increased or decreased by an amount equal to the value of the input carried by that connection.

Block (1962) proved that a simple perceptron can always be taught to separate (classify) linearly separable inputs (Figure 23) without error. Most importantly, he proved that the learning can be accomplished in a finite number of learning trials regardless of the initial values of the weights[5]. This "Perceptron Convergence Theorem" was the first proof that a brain model could actually be "taught" a concrete task (i.e., binary classification).

The Widrow-Hoff Learning Law

At about the same time that Rosenblatt was developing the perceptron, Widrow and Hoff (1960) developed a similar "adaptive pattern classification machine" they called an ADALINE (for ADAptive LINEar classifier). The ADALINE forms an affine combination of the inputs and the weights and passes

---

[5] The only exception to this is if the initial weights are all zero, in which case the network will not train at all.

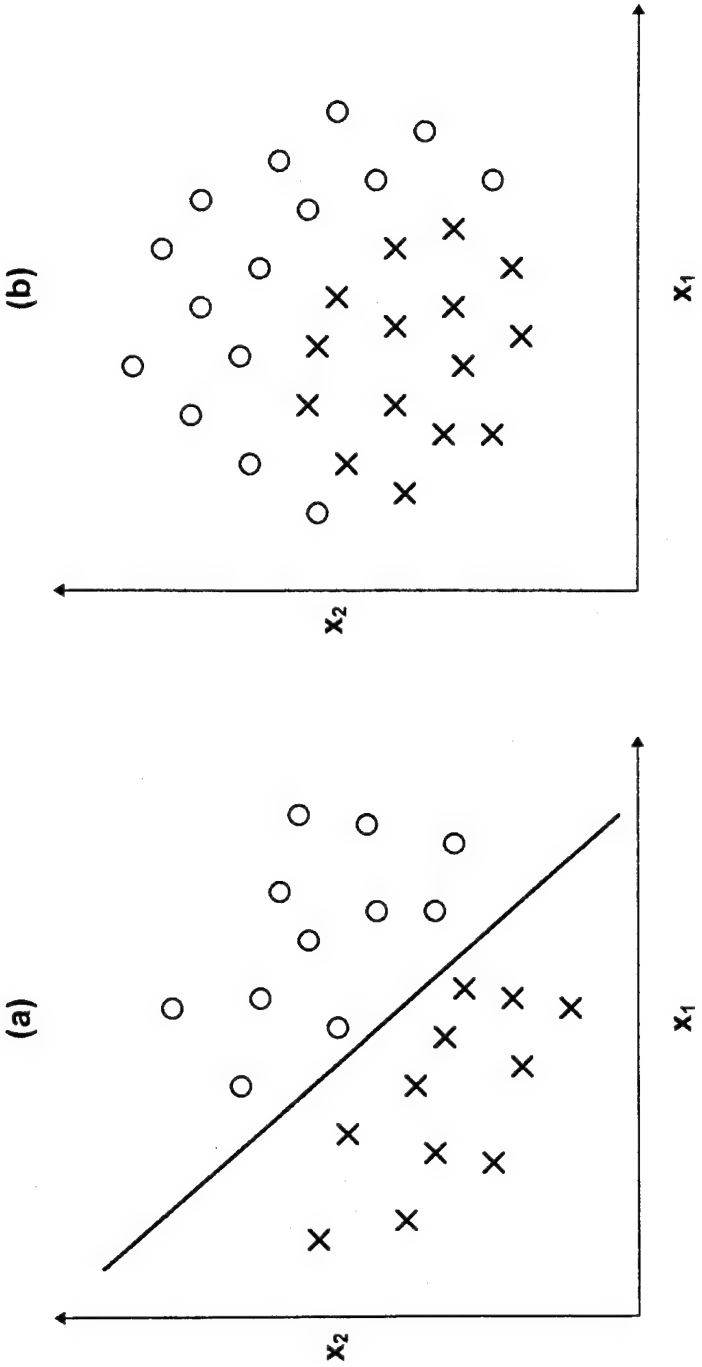


Figure 23. Examples of (a) linearly-separable and (b) linearly-inseparable data

it to a "quantizer" that outputs a +1 if the result is greater than zero and a −1 if the result is less than zero (Figure 24). Mathematically, this can be modeled as

$$y_j = \mathrm{sgn}(s_j) \tag{5}$$

where s_j is defined as before.

On the surface, the ADALINE sounds remarkably like a perceptron. The crucial distinction lies in the learning law used in its training. Perceptrons are essentially trained by adjusting their weights so that the affine combination of the inputs and the weights is either greater than or less than zero. In contrast, the goal of the Widrow-Hoff Learning Law (also called the Delta Rule for reasons which will become apparent later) is to adjust the weights so the affine combination of the inputs and the weights is *identically equal* to either +1 or −1. (This essentially eliminates the need for the quantizer when training is complete.) Most importantly, the ADALINE weights are changed whenever the magnitude of s_j is not identically equal to one, even though the classification (the sign of s_j) may be correct. This offers a vast improvement over the perceptron, whose weights are only changed when the answer is wrong. As the perceptron approaches 100% accuracy, it produces fewer and fewer wrong answers, so its weights are updated less and less frequently. Therefore, it learns ever more slowly as its accuracy improves. The ADALINE, on the other

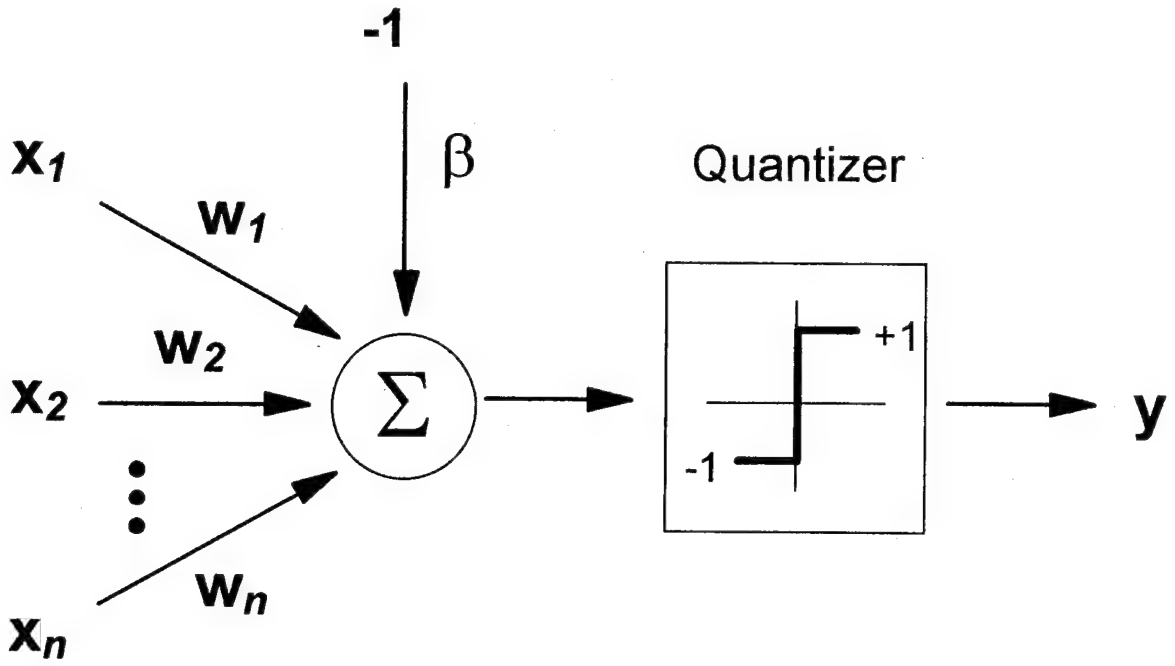


Figure 24. Schematic drawing of an ADALINE neuron

hand, is continually learning (and improving its accuracy) with every example it is shown.

Mathematically, the training goal of the Widrow-Hoff learning law is to find the set of weights **w** that minimizes the sum squared error of the ADALINE over all of the exemplars in the training set:

$$E(\mathbf{w}) = \sum_k E^k \tag{6}$$

where

$$E^k = \frac{1}{2}\sum_j \left(t_j^k - s_j^k\right)^2 \tag{7}$$

Here, the superscript k denotes the individual exemplars from the training set and the subscript j denotes the individual neurons in the network. The constant ½ is introduced purely for mathematical expediency and does not affect the final weights in the trained network. Note, too, that s_j is considered the output of the ADALINE because the quantizer is ignored during training.

Equations 6 and 7 describe an "error surface" in some multidimensional "weight space". The error surface is clearly a quadratic function of the weights; therefore, it has a global minimum value that can be found numerically using, for example, the method of steepest descent.

The method of steepest descent seeks the global minimum point on a multidimensional surface by incrementally "stepping" in the direction in which the surface descends most steeply. Specifically, each ADALINE weight w_{ij} is adjusted by an incremental amount Δw_{ij} that is proportional to the instantaneous slope (gradient) of the error surface with respect to that weight:

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}} = \sum_k \left(-\frac{\partial E^k}{\partial w_{ij}} \right) \tag{8}$$

Using the chain rule of differentiation, $-\partial E^k / \partial w_{ij}$ can be written as the product of two partial derivatives:

$$-\frac{\partial E^k}{\partial w_{ij}} = -\frac{\partial E^k}{\partial s_j^k} \frac{\partial s_j^k}{\partial w_{ij}} \tag{9}$$

The first determines how the error changes as a function of changing the output and the second determines how the output changes as a function of changing the weights. From Equation 2, it is clear that

$$\frac{\partial s_j^k}{\partial w_{ij}} = x_i^k \tag{10}$$

and, by defining

$$\delta_j^k = -\frac{\partial E^k}{\partial s_j^k} = -\left(t_j^k - s_j^k\right) \tag{11}$$

(from which the name "Delta Rule" is obtained), Equation 8 can be rewritten as

$$\Delta w_{ij} \propto \sum_k \delta_j^k x_i^k \tag{12}$$

Introducing the constant of proportionality η, Equation 12 can be rewritten as

$$\Delta w_{ij} = \eta \sum_k \delta_j^k x_i^k \tag{13}$$

The constant η is called the "learning rate". It controls the size of the step taken down the hill. If the step size is too small, a large number of iterations will be required to reach the global minimum. Furthermore, there will be a tendency for the solution to get stuck at the bottom of small local minima. If, on the other hand, η is too large, the gradient descent may overshoot the global minimum. This can lead to a solution that oscillates about the global minimum without ever settling into it.

Equation 13 provides a method for updating the individual weights at the conclusion of each training epoch (i.e., each pass through the entire training set). This is the so-called "batch" method of training. There is an alternative

training method, called "pattern-by-pattern training" in which the weights are adjusted after each exemplar is shown to the network. In that case, the learning law is simply

$$\Delta w_{ij} = \eta \delta_j^k x_i^k \tag{14}$$

Theoretically, the final weights should be identical, regardless of the method chosen. In practice, one method generally works better than the other for a given training set. For example, if the training set is large, training may be slowed by only updating the weights once per training epoch. Conversely, if the training set is small, updating the weights after each exemplar can cause the gradient descent algorithm to meander all over the error surface. Depending on the shape of the error surface, this may prevent it from ever settling into the global minimum.

One very important point about the Widrow-Hoff learning law is that it operates entirely within a single neuron. Each neuron in the network adjusts its own weights based only on the current values of its inputs and its output. There is no need to store any information other than the current value of the weight and no need to know anything about the other neurons in the network. This idea of localization is very important from the standpoint of biological plausibility; however, it is also important if one wishes to implement artificial

neural networks on parallel computers. Because data transmission between nodes in a parallel computer is always a bottleneck to be overcome, a training scheme that does not rely on outside information is extremely desirable.

The idea of localization is also important because it allows a network of ADALINE units (called a MADALINE, which stands for Many ADALINEs) to be assembled in parallel (Figure 25). Since each neuron needs only worry about its own inputs and outputs, training is no more difficult than for a single unit. Now, however, a pattern of inputs can be taught to produce a pattern of outputs. This goes beyond mere pattern recognition—it provides a rudimentary ability to perform function mapping.

The Dark Ages of Neurocomputing

Unfortunately, the processing capabilities of perceptron networks proved to be just that—rudimentary. As more and more research was performed on perceptrons and ADALINEs and dozens of their variants, it became increasingly clear that they all had some major limitations.

In the book *Perceptrons*, Minsky and Papert (1969) developed a logical proof that perceptron networks were severely constrained in their computing abilities. Among their proofs, they showed that perceptrons could not compute even a simple *n*-bit parity (i.e., whether the number of active inputs out of a total of *n* inputs is odd or even). The simplest *n*-bit parity problem, the 2-bit parity problem, is nothing more than the logical predicate XOR (exclusive OR):

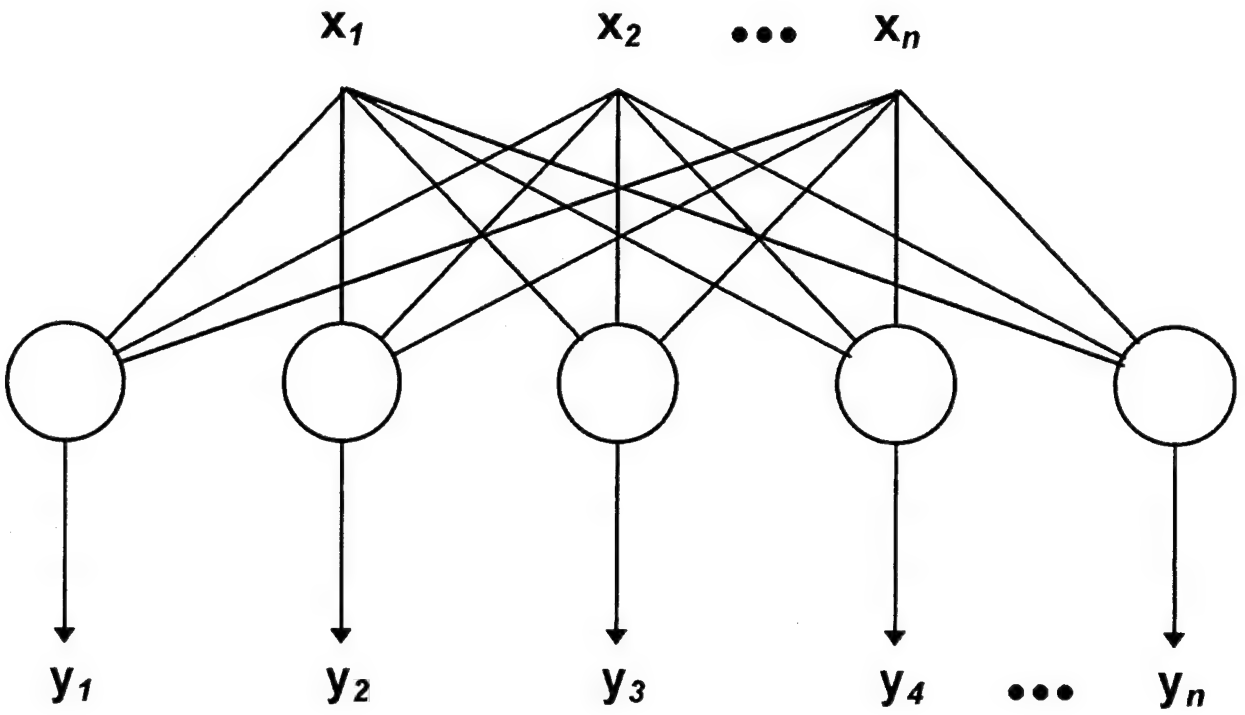


Figure 25. Schematic drawing of a MADALINE (an ADALINE network)

if one or the other of the inputs is active, but not both, the parity is odd; otherwise it is even. This is the simplest logical predicate that is *not* linearly separable and has become a standard test for neural network algorithms.

In addition to proving that single-layer perceptron networks had limited abilities, the authors, in the last chapter of their book, conjectured that the same limitations would extend to multi-layer perceptron networks as well. This conclusion effectively killed artificial neural network research for the next 15 years: if more complex problems could not be solved, even with multi-layer networks (which no one at the time knew how to train anyway), there was no sense in funding further neural network development.

Fortunately, some work in artificial neural networks did continue—mostly as tools for understanding memory. Of particular note was work done by Kohonen (1972) and Anderson (1972) who developed associative memory models[6] based on neurons with real-valued inputs and outputs instead of the binary values used in the perceptron and the ADALINE. These "analog" neurons produced an output proportional to the weighted sum of their inputs

---

[6] In an associative memory model, each input vector is associated with a certain output vector. When properly implemented, even a partial input is sufficient to "recall" the appropriate output. This ability has been used to great advantage in "autoassociative" networks which are taught to map vectors onto themselves; then, if the network is presented with a portion of an input vector, the network "fills in the blanks" to produce the complete vector.

(Figure 26). This was actually quite convenient because it meant that the entire network could be described using simple matrix multiplications. It also meant that the networks could be trained using the Delta Rule, since the analog neurons are operationally identical to an ADALINE with the quantizer removed.

According to Hecht-Nielson (1990), the eventual resurgence of artificial neural networks can be linked directly to Ira Skurnick, a program manager at the Defense Advanced Research Projects Agency (DARPA). In 1983, he provided a small amount of government funding for neurocomputing research—the first since *Perceptrons* was published in 1969. This symbolic step, coupled with the publication of two definitive papers by Hopfield (1982, 1984), re-ignited the field. The second of those papers is especially important in the context of multi-layer, feed-forward neural networks because it presents the first model of a nonlinear neuron.

The Hopfield Neuron

The nonlinear neuron introduced by Hopfield (1984) forms an affine combination of the inputs and the weights, then transforms the result using a nonlinear function to obtain the desired output (Figure 27). Mathematically, this can be expressed as

$$y_j = g(s_j) \tag{15}$$

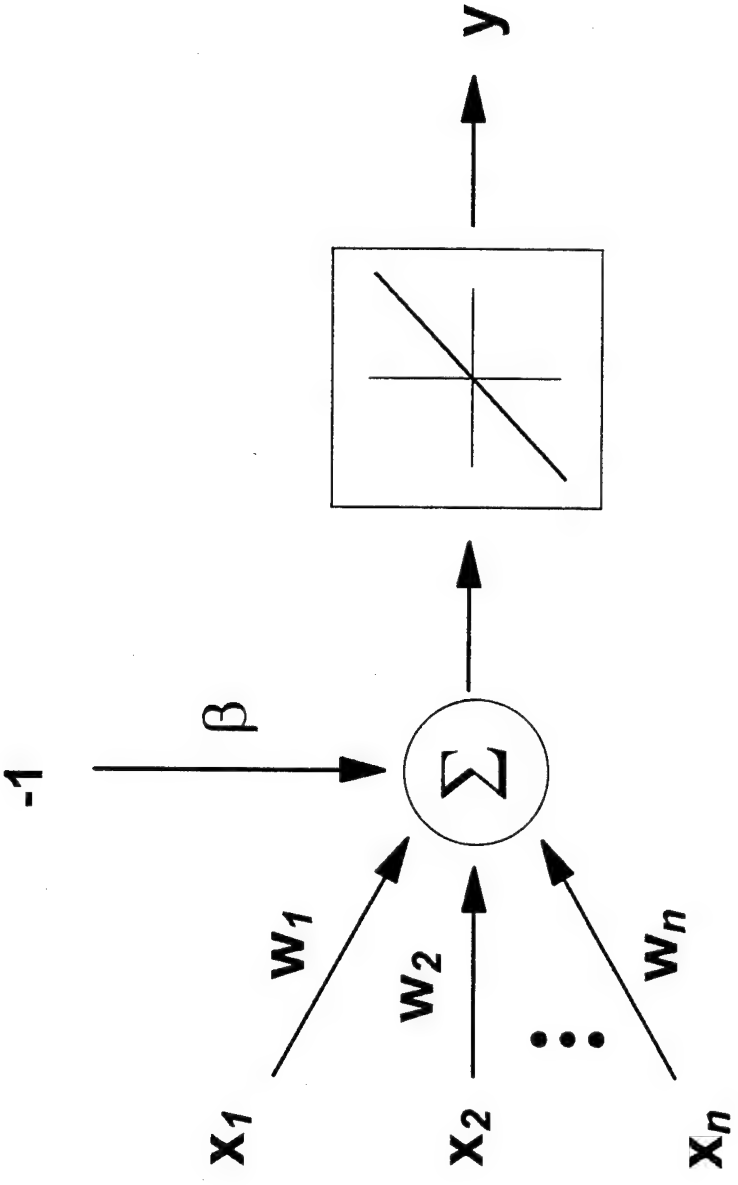


Figure 26. Schematic drawing of a linear analog neuron

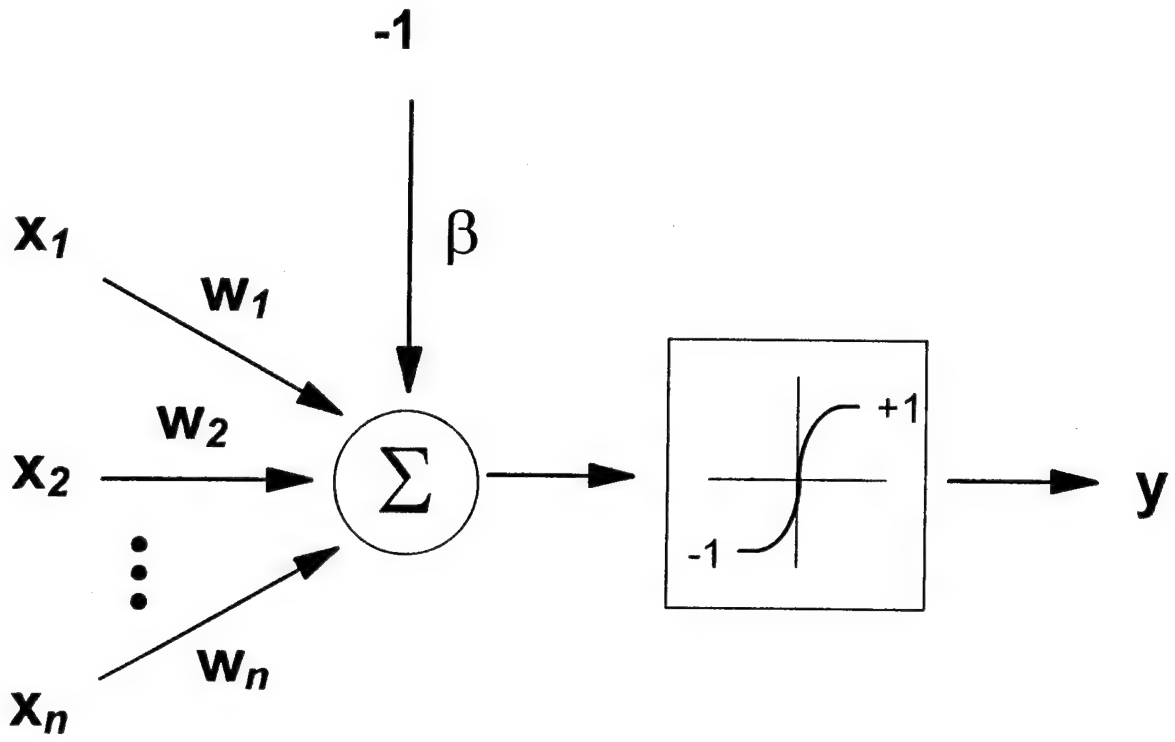


Figure 27. Schematic drawing of a Hopfield neuron

where s_j is defined as before. The function $g(\cdot)$, which is called the transfer function, is usually taken to be the sigmoidal logistic function

$$g(x) = \frac{1}{1+e^{-x}} \tag{16}$$

It should be readily apparent that the mathematical model for the Hopfield neuron is identical to that used for the Perceptron (which Hopfield used in the earlier of the two papers cited), except the Heaviside function has been replaced by a nonlinear, continuously-differentiable function.

This seemingly minor change expanded the capabilities of neural networks by several orders of magnitude. Perceptron networks can only implement certain predicate functions (i.e., binary functions of binary inputs) and networks built from linear neurons were only capable of linear associations in which similar inputs mapped to similar outputs. In contrast, multi-layered networks of nonlinear neurons would be capable of much more complex mappings. Among other things, they could implement mappings that required similar inputs to be mapped onto dissimilar outputs and dissimilar inputs onto similar outputs. The *n*-bit parity problem is just such a mapping. Two *n*-bit patterns that differ by a single bit (and are therefore very similar) map onto outputs that are the exact opposite of one another. Similarly, two patterns that differ in every single bit will map onto the exact same output if *n* is even.

Unfortunately, there were no methods available for training multi-layered networks. The Delta Rule only works for single-layer networks because it requires that the output error for each neuron be known *a priori*. This is easily accomplished in a single-layer network by using supervised learning. In contrast, multi-layer networks have one or more "hidden" layers (Figure 28) separating the network inputs from the output layer. It is not at all apparent what the outputs of the neurons in those hidden layers must be in order to obtain a correct value at the output layer.

Error Backpropagation

Rumelhart, Hinton, and Williams (1986) finally overcame this stumbling block by developing a learning algorithm known today as *backpropagation*. Amazingly, Le Cun (1986) and Parker (1986) simultaneously arrived at the same solution and, as it turns out, Werbos (1974) had derived the same algorithm as part of his doctoral dissertation.

The backpropagation algorithm was developed for a fully-connected, multi-layer, feed-forward network with nonlinear neurons of the type developed by Hopfield. In such a network, each neuron 1) accepts one input from each of the neurons in the layer above it, 2) forms an affine combination of those inputs and its own weights, 3) applies the nonlinear transfer function to that affine combination, and 4) makes the result available to every neuron in the layer below it.

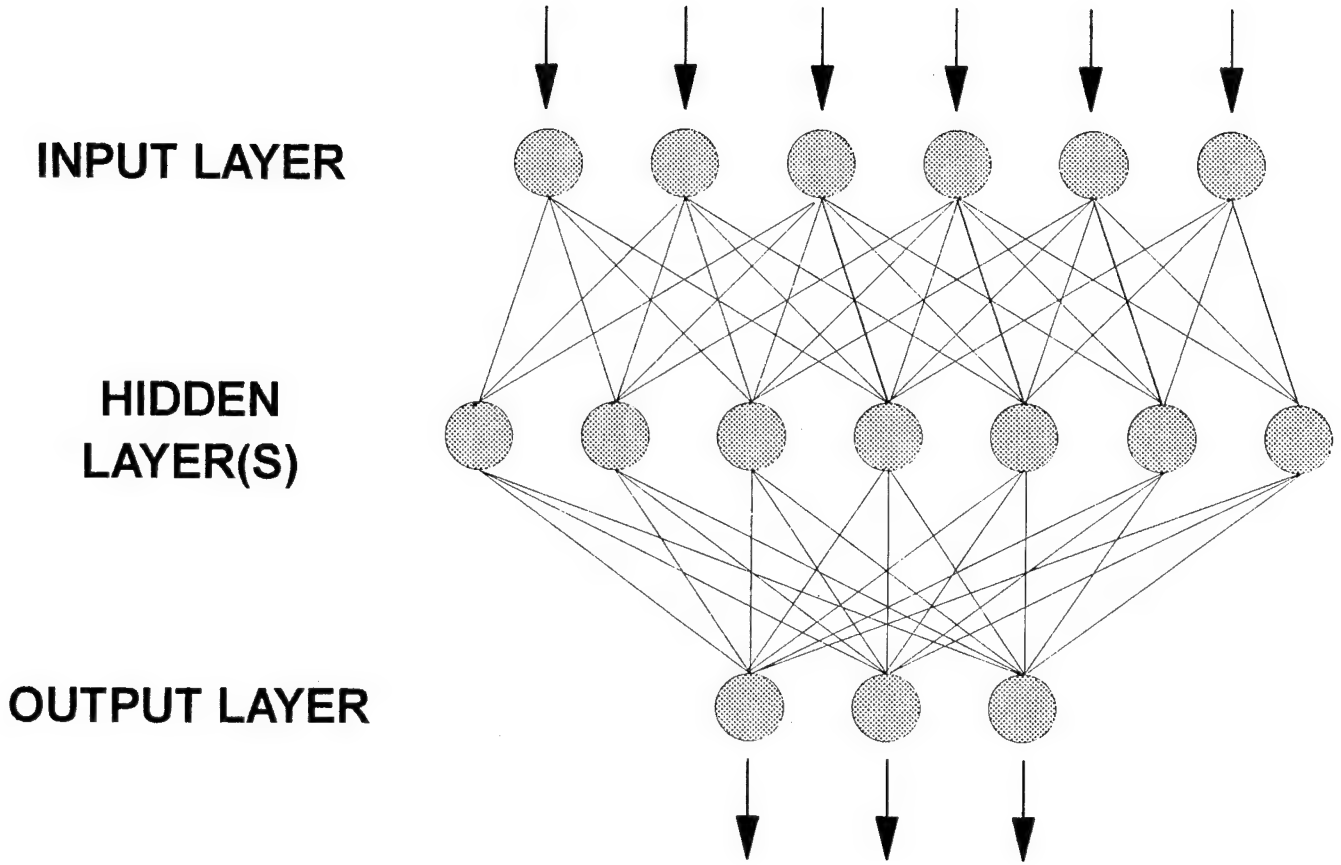


Figure 28. Schematic drawing of a typical multi-layer feed-forward network

The backpropagation algorithm (also called the "generalized delta rule") is based on the original Widrow-Hoff algorithm. It provides a way to adjust the weights of hidden neurons based on the error at the output layer. It does this by propagating the output errors backward through the network. (Hence the name.) This backpropagation of the errors tells the hidden neurons two things: 1) how strongly they are connected to each output neuron, and 2) the error at each output neuron. If a hidden neuron is weakly connected to an output neuron with a large error or strongly connected to an output neuron with a small error, it should not have to modify its weights substantially. Conversely, if a hidden neuron is strongly connected to an output neuron with a large error, it should bear the brunt of the modifications needed to correct that error.

The derivation of the generalized delta rule begins with the gradient descent definition used by Widrow and Hoff:

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}} = \sum_k \left(-\frac{\partial E^k}{\partial w_{ij}} \right) \tag{17}$$

This time, however,

$$E^k = \frac{1}{2}\sum_j \left(t_j^k - y_j^k\right)^2 = \frac{1}{2}\sum_j \left[t_j^k - g\left(s_j^k\right)\right]^2 \tag{18}$$

because each neuron's output is a nonlinear function of s_j rather than s_j itself.

Using the chain rule of differentiation, $-\partial E^k/\partial w_{ij}$ must now be written as the product of three partial derivatives:

$$-\frac{\partial E^k}{\partial w_{ij}} = -\frac{\partial E^k}{\partial y_j^k}\frac{\partial y_j^k}{\partial s_j^k}\frac{\partial s_j^k}{\partial w_{ij}} \tag{19}$$

The first determines how the error changes as a function of changing the output, the second determines how the output changes as a function of changing the affine combination of the weights and the inputs, and the third determines how that affine combination changes as a function of changing the weights. This last term is, as before, simply

$$\frac{\partial s_j^k}{\partial w_{ij}} = x_i^k \tag{20}$$

and, by defining the "delta" as

$$\delta_j^k = -\frac{\partial E^k}{\partial y_j^k}\frac{\partial y_j^k}{\partial s_j^k} \tag{21}$$

a weight update rule that is identical to the original Delta Rule is obtained:

$$\Delta w_{ij} = \eta \sum_k \delta_j^k x_i^k \tag{22}$$

It should be immediately obvious that

$$\frac{\partial y_j^k}{\partial s_j^k} = g'\left(s_j^k\right) \tag{23}$$

which is simply the instantaneous slope of the transfer function evaluated at the current value of s_j. This applies to all neurons, regardless of whether or not they are hidden. Now, all that remains to be determined is the $\partial E^k / \partial y_j^k$ for the neurons in the output layer and in the hidden layer(s).

In the output layer of the network, $\partial E^k / \partial y_j^k$ (the partial derivative of the output error with respect to the output) is simply

$$\frac{\partial E^k}{\partial y_j^k} = -\left(t_j^k - y_j^k\right) \tag{24}$$

which follows directly from Equation 18. Thus, in the output layer

$$\delta_j^k = \left(t_j^k - y_j^k\right) g'\left(s_j^k\right) \tag{25}$$

In the hidden layer, the evaluation of $\partial E^k / \partial y_j^k$ is less direct and far less obvious. Now, the goal is to determine the change in the *output layer* outputs that results from a change in the *hidden layer* outputs. Applying the chain rule of differentiation:

$$\frac{\partial E^k}{\partial y_j^k} = \sum_m \frac{\partial E^k}{\partial s_m^k} \frac{\partial s_m^k}{\partial y_j^k} \tag{26}$$

where the subscript m denotes a neuron in the output layer and the subscript j now denotes a neuron in the hidden layer. Notice that, because the output y_j of a neuron j in the hidden layer is distributed as an input to *every* neuron in the output layer, the effect of a change in y_j on the total output error can only be found by summing errors over all of the neurons in the output layer.

The term $\partial s_m^k / \partial y_j^k$ can be evaluated by examining Figure 28 and the definition of s_m (Equation 2). Each input x_j to a neuron m in the output layer is the output y_j of a neuron j in the preceding (hidden) layer. Furthermore, each input x_j increases the value of s_m by an amount equal to $w_{jm}x_j$. Therefore,

$$\frac{\partial s_m^k}{\partial y_j^k} = w_{jm} \tag{27}$$

Next, note that, by definition,

$$\frac{\partial E^k}{\partial s_m^k} = \delta_m^k \tag{28}$$

which is simply the "delta" used to train the neurons in the output layer. Substituting Equations 27 and 28 into Equation 26,

$$\frac{\partial E^k}{\partial y_j^k} = \sum_m \delta_m^k w_{jm} \tag{29}$$

and, substituting this result into Equation 21,

$$\delta_j^k = g'\left(s_j^k\right)\sum_m \delta_m^k w_{jm} \tag{30}$$

Equation 30 defines a recursive relationship for computing the deltas in *any* hidden layer as a function of the deltas in the layer immediately succeeding it. The deltas in the output layer are known *a priori*, so the calculations begin there and work backwards toward the input layer.

Error Backpropagation with Momentum

Recall from the discussion of ADALINEs that neural network training is essentially a search for the global minimum of a network error surface in some multi-dimensional weight space. For a single ADALINE, the error surface is a quadratic function of the weights with a clearly-defined minimum. The error surface for a multi-layered neural network, on the other hand, is much more convoluted; it can have very many *local minima* whose error levels are greater than the global minimum error that is being sought (Figure 29). The standard gradient descent technique can produce a search path that settles into a local minimum and cannot escape. This is especially true if the learning rate is small.

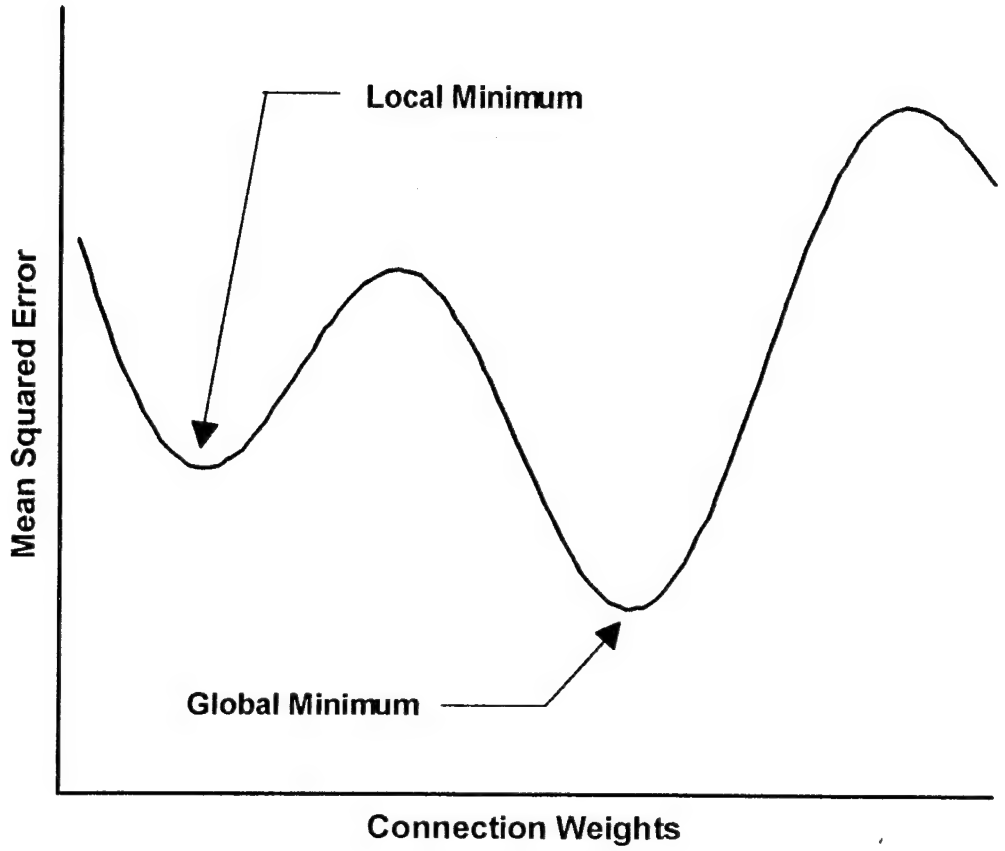


Figure 29. Cross-section of a hypothetical error surface in weight space

The problem cannot be alleviated by simply using a larger learning rate, however, because that can lead to other pathologies such as a search path that oscillates about the global minimum rather than settling into it.

A common technique for ameliorating the problems associated with local minima is to add a momentum term onto the learning rule:

$$\Delta w_{ij}^{new} = \eta \delta_j^k x_i^k + \beta \Delta w_{ij}^{old} \tag{31}$$

Here β is a fractional "momentum factor" that causes a portion of the previous weight change to be reapplied during the current weight update. This serves to keep the weight changes going in the same direction—hence the name.

The addition of a momentum term actually serves several purposes. By itself, the added forward momentum can prevent the search from settling backwards into a local depression (Figure 30). The addition of momentum can also serve as a damper that prevents the search from oscillating endlessly about the global minimum (Figure 31). This allows a higher learning rate to be used which, in turn, lessens the possibility of entering the narrower local depressions. Finally, because each successive weight change contains a portion of its predecessor, the gradient search is imbued with a "memory" of sorts. Each weight change becomes a moving average of the past weight changes, with the most recent figuring more heavily into the average. In fact, if

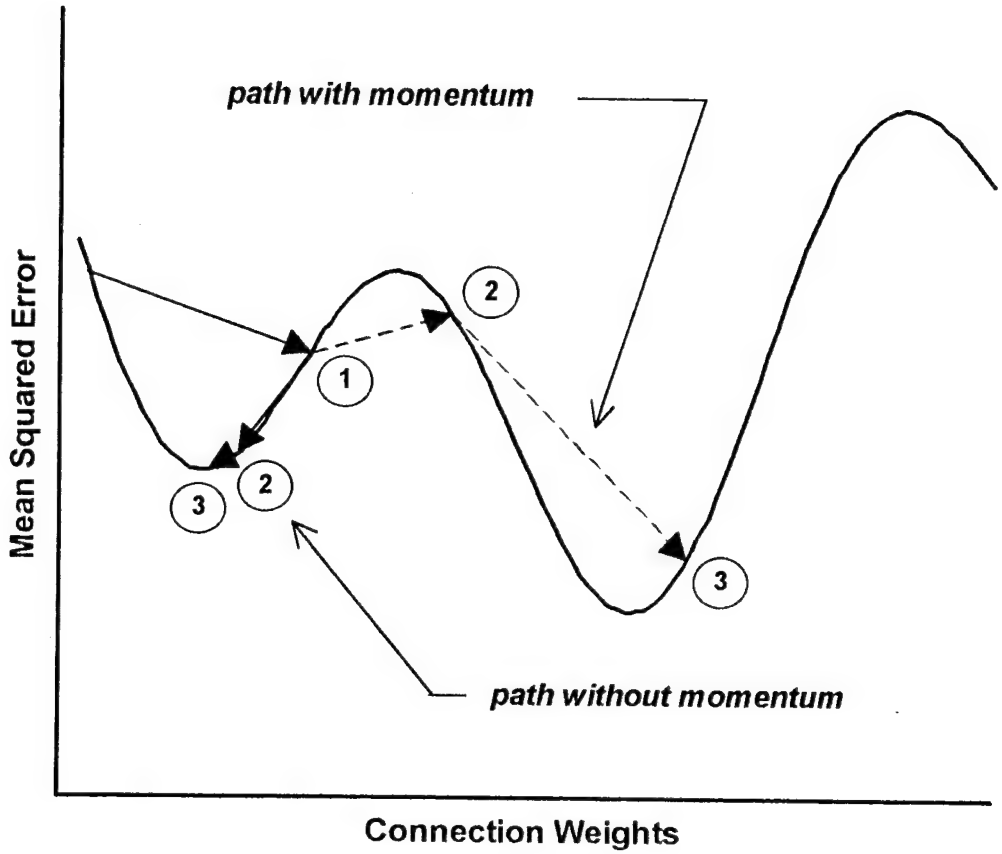


Figure 30. Role of momentum in escaping local depressions

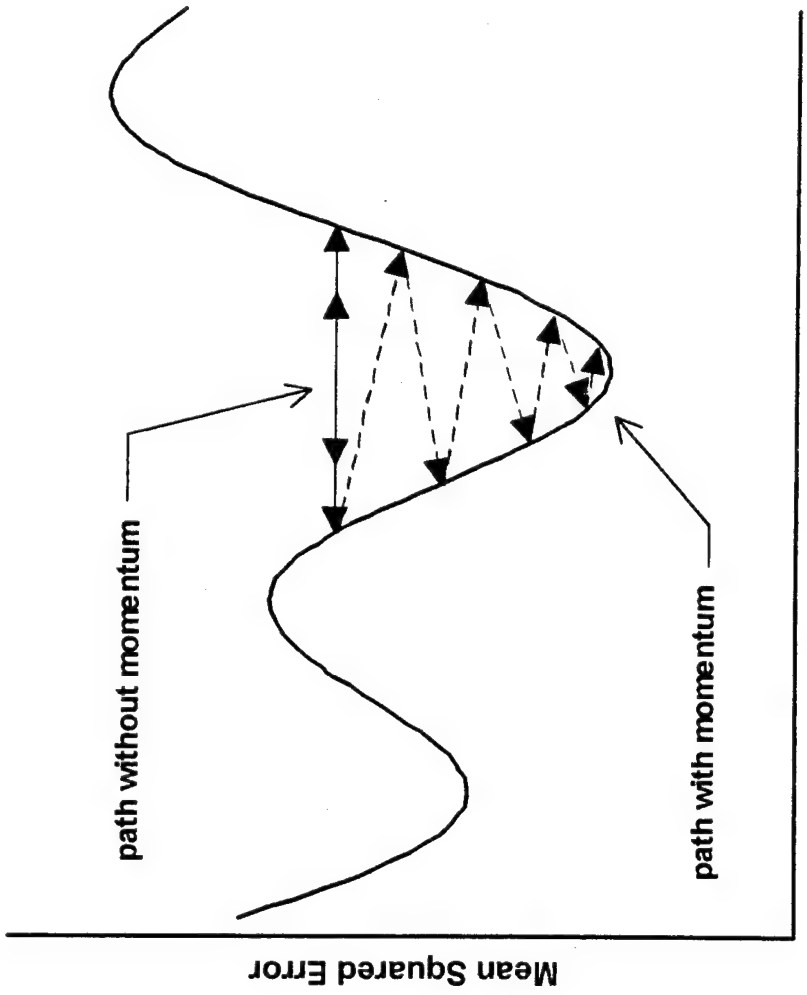


Figure 31. Role of momentum in avoiding oscillations

the momentum factor is identically equal to one, each weight change is an arithmetic average of all the past weight changes. Hagiwara (1992) showed that this is equivalent to defining the error function as the sum of the output errors produced over the entire training set, rather than for the current exemplar. The "memory" obtained with a momentum term can minimize some of the meandering inherent in pattern-by-pattern training. This is especially true at the start of training, when the correlation between the weights and the output errors is much less certain.

Summary

This chapter has presented a chronology of the ideas that have led to the development of multi-layer, feed-forward, artificial neural networks such as those used in this research. Along the way, the necessary mathematics have also been presented. Most importantly, a method for training artificial neural networks—error backpropagation—has been described in detail. A FORTRAN computer program (Appendix A) has been written to implement pattern-by-pattern backpropagation training with or without momentum. A simple flow-chart of the program is shown in Figure 32.

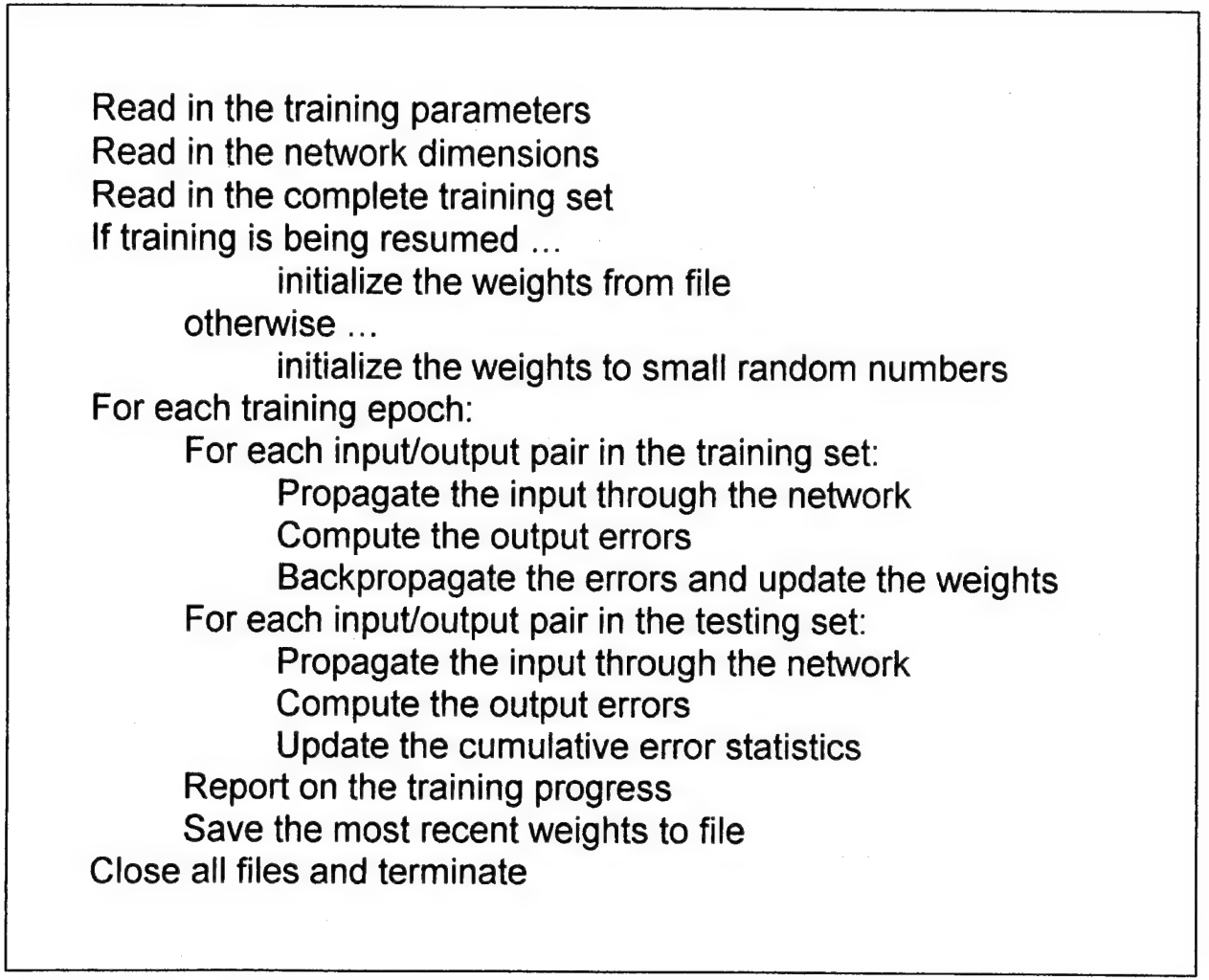


Figure 32. Flowchart for a FORTRAN implementation of backpropagation

CHAPTER 4

SYNTHESIS OF FWD BASINS USING STATIC PAVEMENT ANALYSIS

As mentioned in Chapter 1, the goal of the first half of the research was to show that it is possible to train an artificial neural network to perform real-time backcalculation of pavement layer moduli. In order to accomplish that objective, it was important to be able to obtain a direct comparison of speed and accuracy between the artificial neural network and a conventional backcalculation program. With that in mind, the static, layered-elastic analysis program WESLEA was chosen for generating the synthetic deflection basins needed to train the network.

As was discussed in Chapter 2, WESLEA provides the deflection basin calculations for the iterative, basin-matching inversion program WESDEF. By using WESLEA to generate the neural network training set as well, a direct comparison could be made between what is essentially an artificial neural network implementation of WESLEA and the conventional gradient-descent implementation of WESLEA. If the training of the artificial neural network was successful, it would provide backcalculated moduli that were nearly identical to those obtained using WESDEF—but in much less time.

This chapter will briefly describe the mathematical basis of WESLEA and present the details of the training set generation. The details of the neural network training will be presented in Chapter 6 along with the comparison between the answers obtained with the trained neural network and those obtained using WESDEF. Chapter 6 will also illustrate the differences in speed between the two techniques.

Background

The calculation of the pavement deflections resulting from an FWD test is usually accomplished by analyzing the stresses and displacements induced in a layered elastic half-space by a uniform circular load applied at the surface. This problem was first solved by Burmister (1943) for a two-layered system consisting of a reinforcing layer of finite thickness overlying a half-space. Refinements to his analysis were presented in a series of follow-up papers which gave solutions for the problems of a two-layered system with a frictionless interface between the layers (Burmister, 1945a), a two-layered system with a fully-frictional interface between the layers (Burmister, 1945b), and a three-layered system with fully-frictional interfaces (Burmister, 1945c). This last analysis was important in that it allowed the pavement and the base course to be modeled as separate layers.

DeJong, Peutz, and Korswagen (1973) refined the basic multi-layer analysis to allow a continuum of interface conditions varying between the limits

of frictionless and fully-frictional. Van Cauwelaert, Delaunois, and Beaudoint (1987) went one step further by developing interface conditions that satisfied Coulomb's law. Their work forms the theoretical basis of WESLEA.

Stresses and Displacements in a Layered Elastic Medium

The development of the basic theory of stresses and displacements in a layered elastic medium proceeded from the three-dimensional equations of equilibrium and compatibility. For an axisymmetric problem, force equilibrium in the radial direction is given by

$$\frac{\partial \sigma_r}{\partial r} + \frac{\partial \tau_{rz}}{\partial z} + \frac{\sigma_r - \sigma_\theta}{r} = 0 \tag{32}$$

if inertial forces are neglected. Similarly, force equilibrium in the vertical direction is given by

$$\frac{\partial \tau_{rz}}{\partial r} + \frac{\partial \sigma_z}{\partial z} + \frac{\tau_{rz}}{r} = 0 \tag{33}$$

For small strains, strain compatibility in the radial direction is given by

$$\nabla^2 \sigma_r - \frac{2}{r^2}(\sigma_r - \sigma_\theta) + \frac{1}{1+\nu}\frac{\partial^2}{\partial r^2}(\sigma_r + \sigma_\theta + \sigma_z) = 0 \tag{34}$$

and strain compatibility in the circumferential direction is given by

$$\nabla^2\sigma_\theta + \frac{2}{r^2}(\sigma_r - \sigma_\theta) + \frac{1}{1+\nu}\left(\frac{1}{r}\right)\frac{\partial}{\partial r}(\sigma_r + \sigma_\theta + \sigma_z) = 0 \qquad (35)$$

where ν is Poisson's ratio.

The equations of equilibrium are satisfied by expressing the normal and tangential stresses in terms of a stress function ϕ as follows:

$$\sigma_z = \frac{\partial}{\partial z}\left[(2-\nu)\nabla^2\phi - \frac{\partial^2\phi}{\partial z^2}\right] \qquad (36)$$

$$\sigma_r = \frac{\partial}{\partial z}\left[\nu\nabla^2\phi - \frac{\partial^2\phi}{\partial r^2}\right] \qquad (37)$$

$$\sigma_\theta = \frac{\partial}{\partial z}\left[\nu\nabla^2\phi - \frac{1}{r}\frac{\partial\phi}{\partial r}\right] \qquad (38)$$

$$\tau_{rz} = \frac{\partial}{\partial r}\left[(1-\nu)\nabla^2\phi - \frac{\partial^2\phi}{\partial z^2}\right] \qquad (39)$$

Applying Hooke's law, the displacements in the vertical and horizontal direction can similarly be expressed as

$$w = \frac{1+\nu}{E}\left[(1-2\nu)\nabla^2\phi + \frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r}\right] \tag{40}$$

and

$$u = -\frac{1+\nu}{E}\frac{\partial^2\phi}{\partial r \partial z} \tag{41}$$

respectively, where E is the modulus of elasticity.

The compatibility equations are satisfied if the stress function ϕ is a solution to the biharmonic differential equation

$$\nabla^4\phi = 0 \tag{42}$$

For a uniform surface load of radius R and intensity p, the appropriate stress function (Van Cauwelaert, Delaunois, and Beaudoint, 1987) is:

$$\phi = pR\int_0^\infty \frac{J_o(mr)J_1(mR)}{m}\left[A_i e^{mz} - B_i e^{-mz} + C_i z e^{mz} - D_i z e^{-mz}\right]dm \tag{43}$$

where $J_o(\cdot)$ is a Bessel function of the first kind and order zero, $J_1(\cdot)$ is a Bessel function of the first kind and order one, and the coefficients A_i, B_i, C_i, and D_i are found by satisfying the boundary conditions of each layer.

The stress function given by Equation 43 can be substituted into the equations for the stresses and displacements (Equations 36–41) and evaluated at the layer boundaries to determine the coefficients A_i, B_i, C_i, and D_i for each layer. Among the boundary conditions that must be satisfied are 1) the normal and shearing stresses at the surface of the half-space are zero everywhere outside of the loaded area; 2) all of the stresses and displacements vanish at infinite depth; and 3) continuity of stresses and/or displacements must be satisfied at the layer interfaces (depending on their frictional properties).

The resulting equations are quite complex and must be integrated numerically. Much of the work that has gone into computer programs such as BISAR and WESLEA involved finding ways to efficiently integrate the resulting stress and displacement equations without a loss of accuracy. Because the equations are extremely lengthy, they will not be reproduced here. Instead, the reader is referred to Van Cauwelaert, Delaunois, and Beaudoint (1986) for the details of the stress and displacement equations and to Van Cauwelaert, et. al., (1988) for the details of their computer implementation and the numerical integration techniques applied to their computation.

Training Set Requirements

An artificial neural network learns a multi-dimensional functional mapping through repeated exposure to examples of that mapping. How well it learns that mapping depends as much on the contents of the training set as on

the network topology and training methods. The training set must be designed with the strengths and weaknesses of the network in mind.

Multi-layer, feed-forward artificial neural networks of the type used here (henceforth referred to simply as "backpropagation networks") are very good at generalization[7] but do not extrapolate well. Therefore, the training set should be designed so as to span as much of the expected input space as possible. The training set should also be as representative of the complete input space as possible—if some inputs are statistically more likely than others, the training set should reflect that. (The error being minimized is an average over all of the examples in the training set; those examples that occur most frequently will be fit most precisely when the average error is minimized.) In addition, even though backpropagation neural networks are universal approximators, their training times increase rapidly with increasing problem complexity. This places some practical limits on the complexity of the mappings that can be learned. Finally, care must be taken to present the training examples to the network in a random order. If the backpropagation network is shown repeated examples from only one portion of the input space, it can actually "forget" any previously

---

[7] In the parlance of neural networks, generalization is the ability to perform successfully for input patterns that have never been seen before. In the context of function mapping, this is the same thing as being able to interpolate between the data points that make up the training set.

learned portions. Also, if the training examples are presented in the same order every time, the network may simply trace a loop around the error surface, always returning to the same spot at the conclusion of each epoch. By presenting the examples in random order, more of the weight space is explored while searching for the global minimum, which increases the chances of its being attained.

Statement of the Problem

The requirement that the training set completely cover the input space means that realistic boundaries must be established for the inversion problem. For the initial feasibility study, it was decided that the problem would be restricted to conventional flexible pavement systems consisting of an asphaltic concrete (AC) surface course and an unbound granular base course overlying the subgrade soil.

In the context of the static, layered-elastic analyses used in this portion of the research, the primary variables affecting pavement response are the thicknesses and elastic stiffnesses of the pavement layers. Therefore, the mapping problem that will be taught to the neural network is to map the layer thicknesses and the peak measured deflections onto the backcalculated elastic moduli. This is the same "mapping" that is performed by the traditional backcalculation programs discussed in Chapter 2. Variables such as mass density and Poisson's ratio, which vary over a relatively narrow range in the

pavement materials of interest here, will be treated as constants and set equal to generally accepted (i.e., typical) values.

As was shown in Chapter 2, the thickness of the subgrade (that is, the depth to bedrock) has a significant effect on the peak displacements calculated using a static analysis. On the other hand, it was shown that the thickness of the subgrade has little influence on the deflection basins calculated using a dynamic pavement response model. As a result, the subgrade thickness will not have to be considered in Phase II of the research. To maintain consistency between the two research phases, the subgrade thickness had to be eliminated from consideration in Phase I as well. In the work of Foinquinos, Roesset and Stokoe (1993b), static deflection basins calculated for bedrock depths of 80 ft were almost indistinguishable from those calculated with an infinite depth to bedrock (Figure 18). Chang, et. al. (1992) similarly showed that calculated surface deflections for four completely different pavement systems (ranging from Farm-to-Market roads to Interstate highways) asymptotically approached limiting values as the bedrock depth approached 80–100 ft (Figure 33). The thickness of the subgrade was therefore assigned a value of 100 ft in order to eliminate the influence of the bedrock. Therefore, only two thicknesses (one for the surface layer and the other for the base layer) and three elastic moduli (one each for the surface layer, base layer, and subgrade) are needed to completely characterize the pavement system.

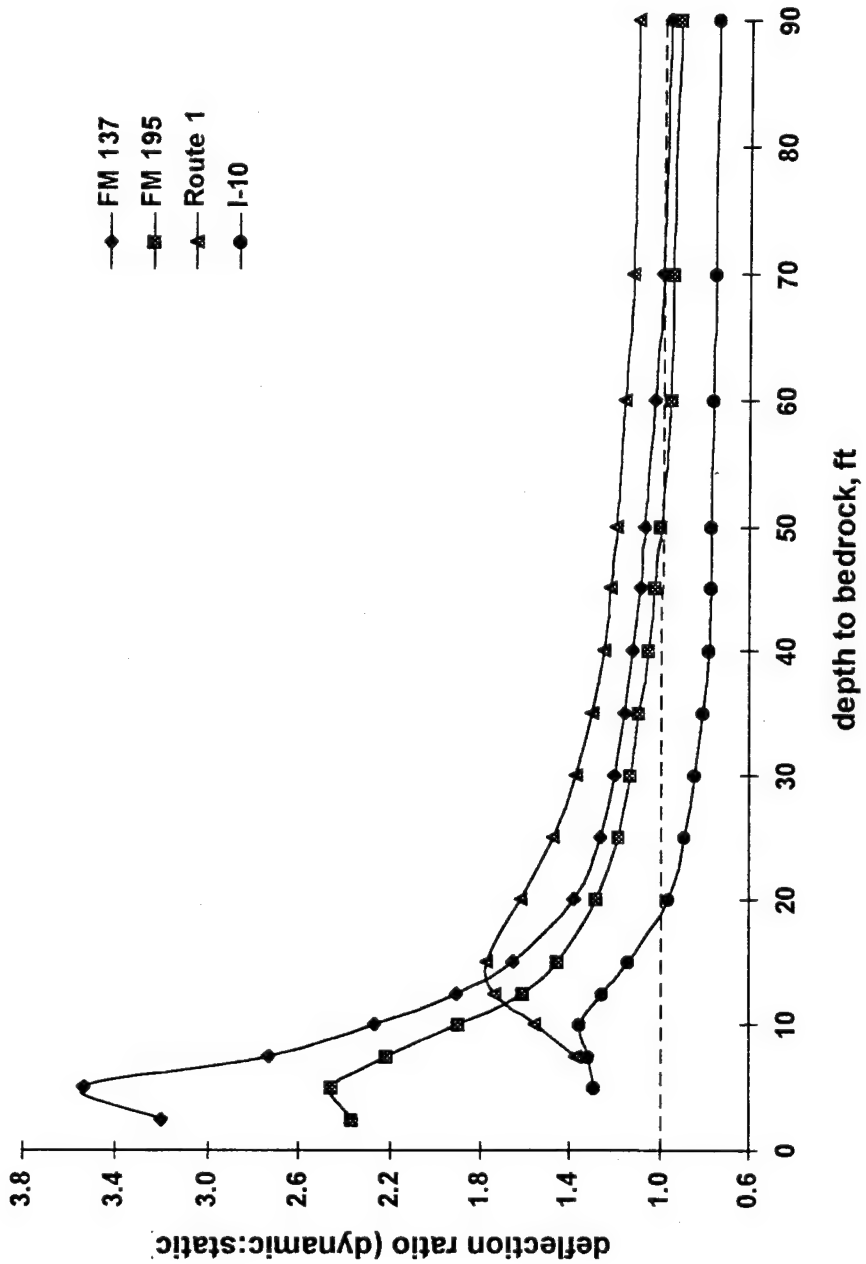


Figure 33. Deflection ratios versus depth to bedrock for four pavement systems (after Chang, et. al., 1992)

Having established the general pavement structure, the next step was to determine the ranges of layer thicknesses and material properties that could be expected for each material layer. SHRP has recently established a standardized backcalculation procedure for its LTPP program (Rada, Richter, and Jordahl, 1994). Among other things, that procedure specifies ranges of layer moduli to be input to the MODULUS backcalculation program in order to bracket the solution. These ranges were adopted as the modulus ranges for the training set. Where ranges were not provided, engineering judgment was used to set the appropriate limits. Table 2 summarizes the ranges of pavement layer properties used to generate the training set.

The final step in defining the mapping problem was to determine the radial offsets at which to output the synthetic measured pavement deflections. As mentioned in Chapter 2, the Dynatest FWD has seven sensors which are usually mounted at 12-in intervals out to a distance of 72 in. The KUAB FWD also has 7 sensors, but their offsets are fixed at 0, 8, 12, 18, 24, 36, and 48 in. The SHRP LTPP specifications, on the other hand, call for sensors positioned at offsets of 0, 8, 12, 18, 24, 36, and 60 in. As shown in Table 3, all three sensor arrangements can be accommodated in a single training set with just nine offsets (0, 8, 12, 18, 24, 36, 48, 60, and 72 in). In this way, several neural networks could be trained, each for use with a different set of sensor spacings, by selecting the appropriate deflections from the database.

Table 2. Ranges of layer properties used in the training set

Layer	Thickness (in)	Young's Modulus (ksi)	Poisson's Ratio
Surface	2–12	250–3000	0.325
Base	6–30	5–150	0.350
Subgrade	∞	5–50	0.350

Table 3. Sensor locations represented in the training set

Device/Specification	Sensor Offsets (in)								
	0	8	12	18	24	36	48	60	72
Standard Dynatest	✓		✓		✓	✓	✓	✓	✓
Standard KUAB	✓	✓	✓	✓	✓	✓	✓		
SHRP LTPP Program	✓	✓	✓	✓	✓	✓		✓	

Training Set Generation

A modified version of WESLEA was implemented on the CRAY Y-MP supercomputer at the Waterways Experiment Station to generate the synthetic deflection basins needed to train the neural network. In order to satisfy the requirements that a) the training set have a probability density similar to that of the actual input space and b) the training exemplars be randomly presented to the network during training, pavement profiles were constructed by randomly selecting layer thicknesses and layer moduli from uniform distributions covering the ranges shown in Table 2. A total of 10,000 pavement profiles were constructed in this manner. With five independent variables (two thicknesses and three moduli), this is similar to establishing a test matrix with 10 levels for each variable (10^5 = 10,000) and using every combination available. That degree of coverage should minimize the distances over which the neural network must interpolate and therefore maximize the accuracy of the backcalculated values. The entire training set of 10,000 deflection basins was generated in approximately 30 min.

By randomly selecting the layer properties from uniform distributions, a training set was created in which each of the pavement layer properties was equally likely. Note that this is *not* the same as a training set in which each of the deflection basins is equally likely. As the stiffness of the pavement system increases, there is much less change in the surface deflections for a given

change in stiffness (mathematically, the derivative of deflection with respect to layer stiffness is asymptotically approaching zero). The training set is therefore skewed toward the shallow deflection basins. Showing the network more examples of shallow deflection basins than of deep deflection basins should help it learn the inversion function better because more of its training time is spent on those examples that are the most difficult to invert accurately.

In order to ensure that the distributions of the pavement layer properties in the training set were suitably uniform, histograms of the layer thicknesses and moduli were constructed (Figures 34–38) and summary statistics were calculated (Table 4). All of the layer properties were first normalized by rescaling them to a range of [0,1]. This was done so the summary statistics could be easily compared to those of a uniform (U[0,1]) distribution. The theoretical mean of the U[0,1] distribution is simply $1/2$ and the theoretical variance is $1/12$. The standard deviation (i.e., the square root of the variance) should therefore be 0.2886. The values shown in Table 4 match those statistics very closely and the histograms do not indicate any significant deviations from the assumption of uniformity.

In order to ensure that the pavement layer property distributions were uncorrelated (e.g., to ensure that low asphalt moduli were not more likely to occur when the base modulus was also low), correlation coefficients were computed for each pair of random variables (Table 5). A correlation coefficient

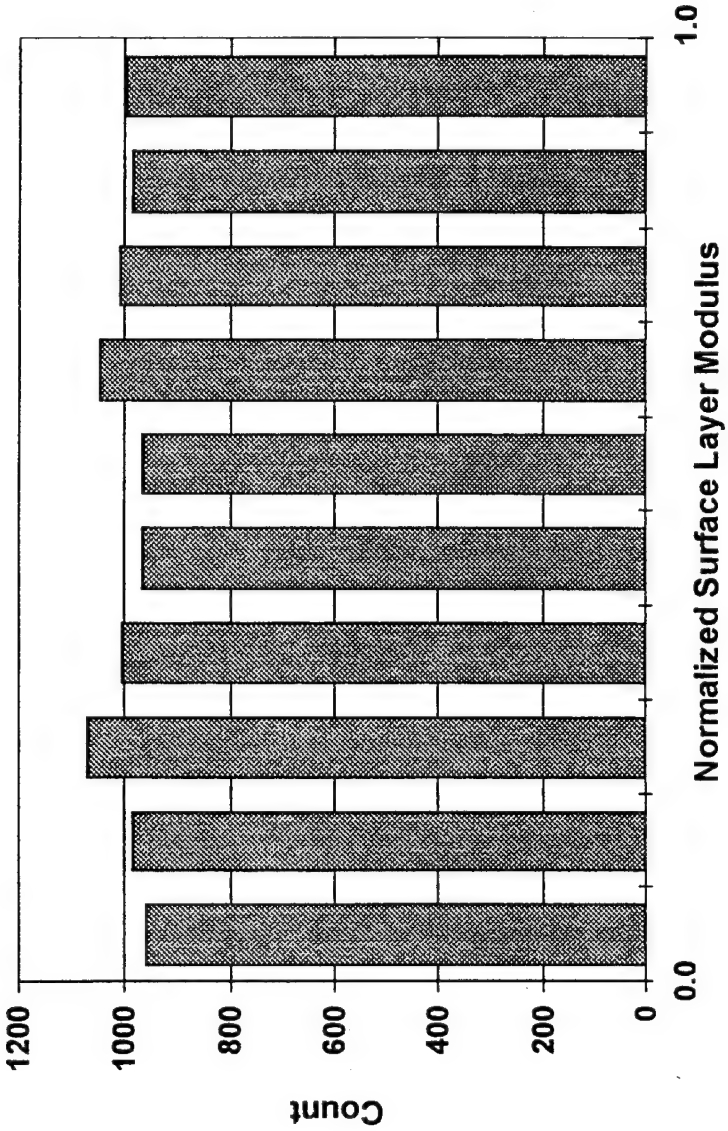


Figure 34. Distribution of surface layer moduli in the training set

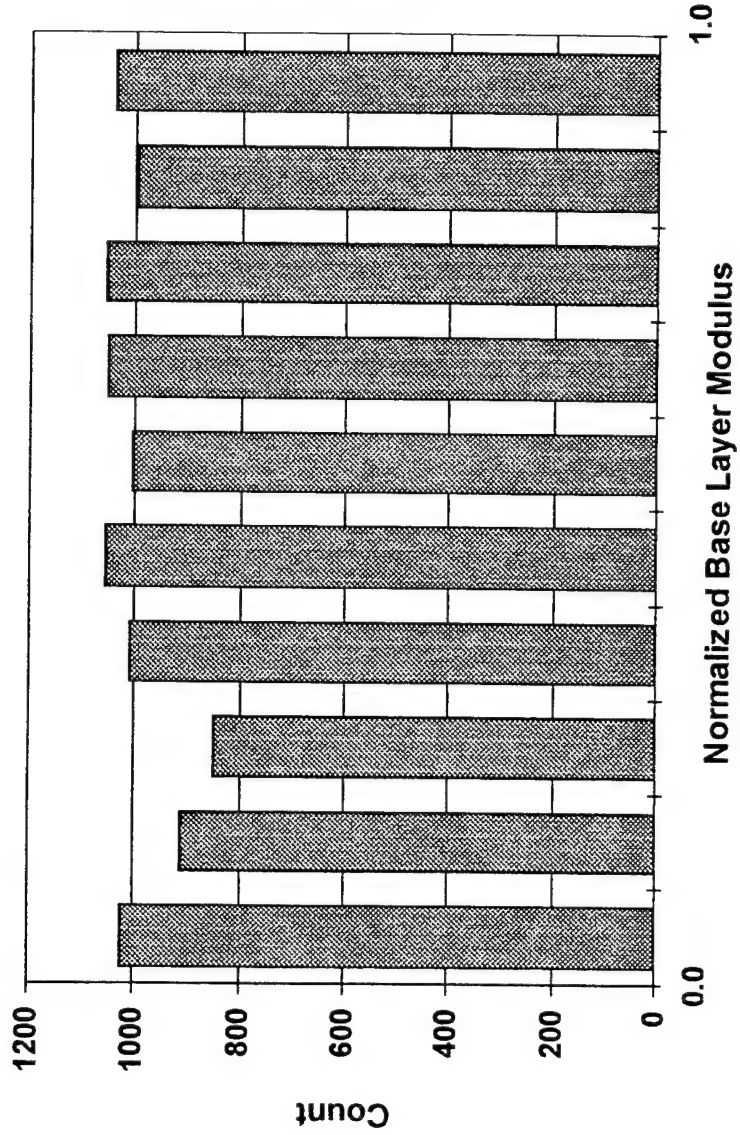


Figure 35. Distribution of base layer moduli in the training set

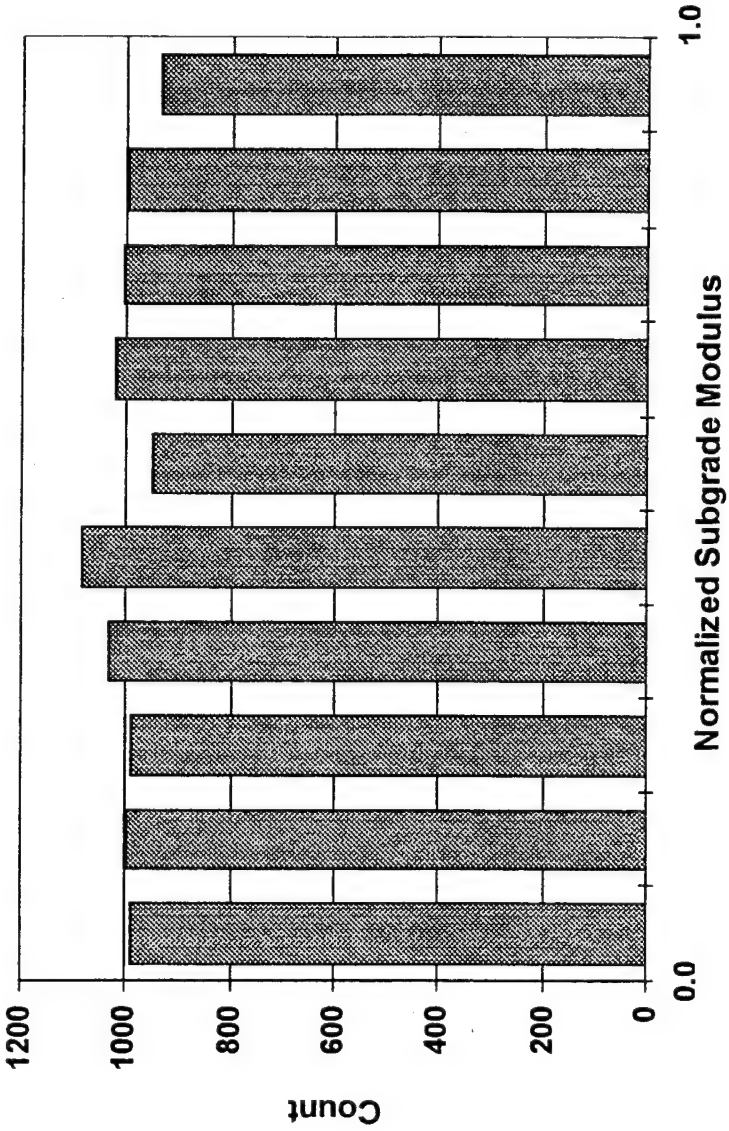


Figure 36. Distribution of subgrade moduli in the training set

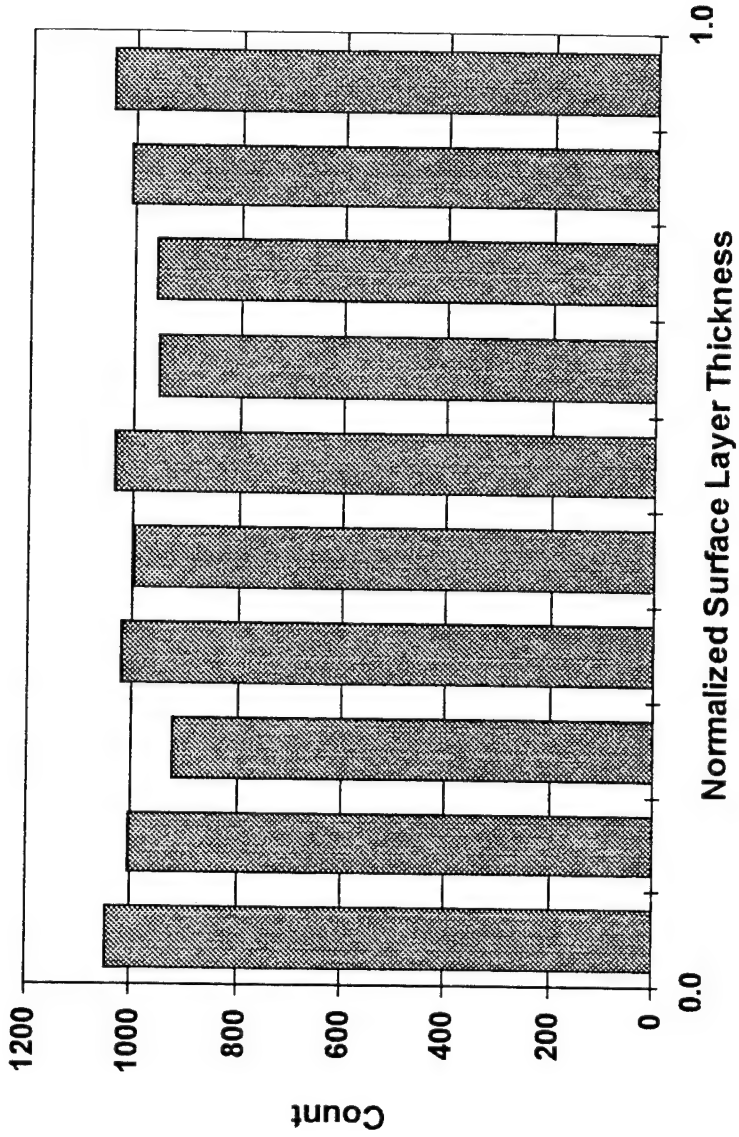


Figure 37. Distribution of surface layer thicknesses in the training set

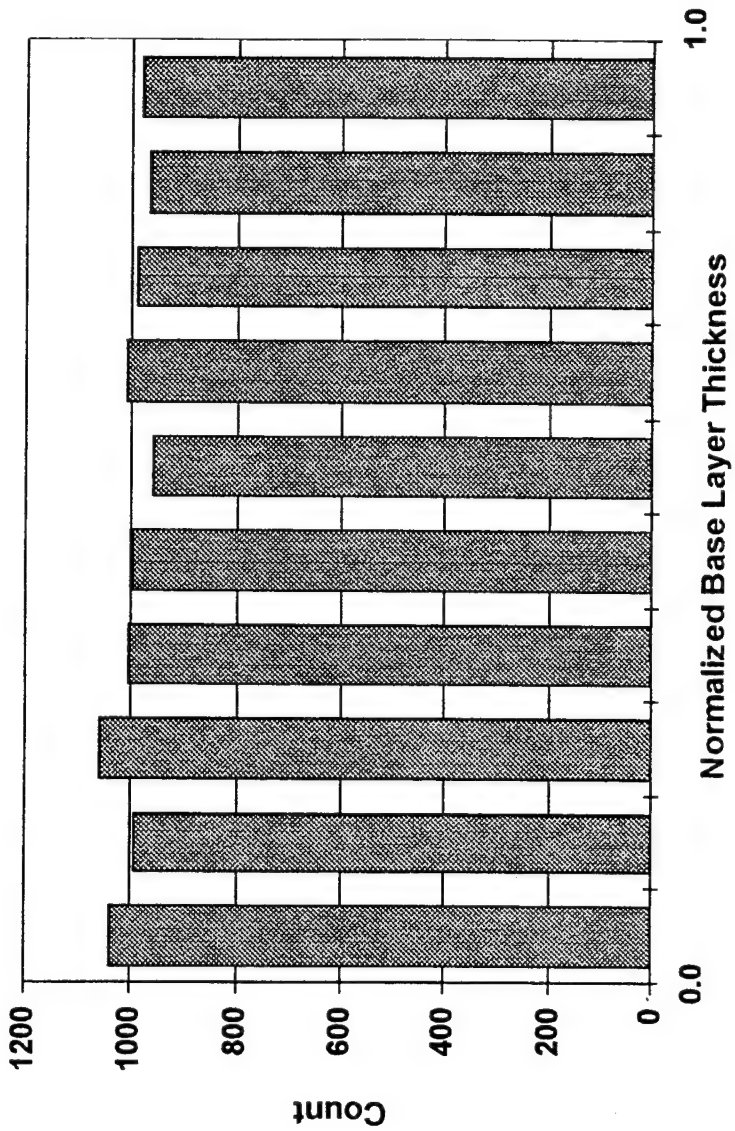


Figure 38. Distribution of base layer thicknesses in the training set

Table 4. Statistics on randomly-generated layer properties

Statistic	Surface Modulus	Base Modulus	Subgrade Modulus	Surface Thickness	Base Thickness
Mean	0.5008	0.5092	0.4971	0.5000	0.4944
Standard Deviation	0.2879	0.2884	0.2862	0.2903	0.2889

Table 5. Correlation coefficients for the layer properties

	Surface Modulus	Base Modulus	Subgrade Modulus	Surface Thickness	Base Thickness
Surface Modulus	+1.0000	−0.0005	+0.0144	+0.0040	+0.0174
Base Modulus	−0.0005	+1.0000	+0.0211	−0.0043	+0.0058
Subgrade Modulus	+0.0144	+0.0211	+1.0000	+0.0042	+0.0017
Surface Thickness	+0.0040	−0.0043	+0.0042	+1.0000	−0.0134
Base Thickness	+0.0174	+0.0058	+0.0017	−0.0134	+1.0000

close to +1 indicates that the variables are linearly dependent on each other (i.e., when one goes up, the other goes up; when one goes down, the other goes down). Similarly, a correlation coefficient close to −1 indicates a negative linear dependence (i.e., as one goes up, the other goes down). A correlation coefficient of zero indicates that the two variables are uncorrelated. All of the off-diagonal correlation coefficients shown in Table 5 are close to zero, so there is no apparent correlation between the variables.

Summary

This chapter has described the creation of a neural network training set consisting of 10,000 synthetic deflection basins. The synthetic deflection basins were calculated for randomly-selected pavement profiles using the static, layered elastic analysis program WESLEA. Because this is the same analysis program that is used in the conventional backcalculation program WESDEF, a direct comparison of accuracy and speed can be made between the trained neural network and WESDEF. This comparison, and the details of the network training, are presented in Chapter 6.

CHAPTER 5

SYNTHESIS OF FWD BASINS USING DYNAMIC PAVEMENT ANALYSIS

The goal of the first phase of the research was to show that it is possible to teach an artificial neural network to backcalculate pavement layer moduli from FWD deflection basins. If successful, the computational efficiency of the neural network would allow it to backcalculate those moduli in real time. As was mentioned in Chapter 2, the computational efficiency of the trained neural network is independent of the computational complexity of the algorithms used to develop its training examples. This feature of neural networks makes it possible to perform real-time backcalculation using much more realistic models of pavement response than are currently used in conventional basin-matching programs. A significant increase in the realism of the pavement response model can be realized by replacing the static analysis afforded by WESLEA with an elastodynamic analysis of the FWD test. This will simultaneously increase the accuracy of the backcalculated moduli and eliminate their dependence on the assumed depth to bedrock.

The goal of the second research phase was therefore to retrain the neural network developed in Phase I using synthetic deflection basins

generated by an elastodynamic analysis of the pavement response. Because the task of mapping deflection basins (and pavement layer thicknesses) onto their corresponding elastic moduli remains essentially the same—only the magnitudes of the deflections will differ from those used in the first phase of the research—there should be no reason to change the network's architecture. By preserving the network architecture, the computational speed of the network will also be preserved. In other words, the retrained network will still be able to backcalculate pavement layer moduli in real time, but will do so based on a dynamic analysis of the pavement's response to the FWD loads rather than a static analysis.

This chapter will briefly describe the elastodynamic model of the FWD test and its numerical implementation and will present the details of the training set generation for the second phase of the research. The retraining of the neural networks using that data set will be discussed in Chapter 6.

Frequency Domain Analysis

A common method for analyzing the response of a linear system to transient loads is by superposition of the system's responses to steady-state excitations at many different frequencies. This method, known alternately as Fourier superposition analysis or frequency domain analysis, is a very flexible tool that can be easily applied to any linear system for which the solution to the steady-state problem is known. For the problem of interest, the steady-state

solution is provided by elastodynamic Green functions, described in the next section, that relate the harmonic response of the pavement surface to a harmonic surface load.

To perform a frequency domain analysis, the transient applied load p(t), which is a continuous function of time, is sampled at integer multiples of a specified time increment Δt to obtain a discretized waveform. That waveform can be represented by the discrete values

$$p_n \equiv p(n\Delta t) \quad , \quad n = 0, 1, \ldots , N-1$$

where N is the number of samples. The discretized waveform is then decomposed into N separate frequency components

$$P_k \equiv P(k\Delta\omega) \quad , \quad k = -\frac{N}{2}, \ldots , 0 , \ldots , \frac{N}{2}-1$$

where

$$\Delta\omega = \frac{2\pi}{N\Delta t} \tag{44}$$

is the corresponding sampling interval in the frequency domain. This decomposition can be accomplished using a discrete Fourier transform:

$$P_k = \Delta t \sum_n p_n e^{-2\pi i k n/N} \tag{45}$$

Note that even though the p_n are real numbers, the P_k are complex numbers whose real and imaginary parts contain information concerning both the magnitude and the phase of the individual frequency components. The magnitudes of the components are given by

$$|P_k| = \sqrt{Re^2(P_k) + Im^2(P_k)} \tag{46}$$

and the phase angles are given by

$$\theta_k = \tan^{-1}\left(\frac{Im(P_k)}{Re(P_k)}\right) \tag{47}$$

where $Re(P_k)$ and $Im(P_k)$ are the real and imaginary parts, respectively, of P_k.

The next step in the analysis is to obtain individual solutions for the displacements produced by a unit harmonic load at each of the frequencies represented. These solutions are known as the fundamental or Green function solutions. They can be represented by the transfer functions:

$$H_k \equiv H(k\Delta\omega) \quad , \; k = -\frac{N}{2}, \ldots, 0, \ldots, \frac{N}{2} - 1$$

These individual, complex-valued transfer functions are multiplied by the individual frequency components of the applied load history to obtain a series of steady-state displacements:

$$U_k \equiv P_k H_k \ , \ k = -\frac{N}{2}, \ldots, 0, \ldots, \frac{N}{2}-1$$

These individual steady-state displacement solutions (which are also complex-valued) actually represent the individual frequency components of a discretized displacement history

$$u_n \equiv u(n\Delta t) \ , \ n = 0, 1, \ldots, N-1$$

that can be recovered from the frequency domain using the inverse discrete Fourier transform:

$$u_n = \frac{\Delta \omega}{2\pi} \sum_k U_k e^{2\pi i k n/N} \qquad (48)$$

Because the p_n were initially real numbers, the resulting u_n will also be real numbers. They are a discrete representation of a continuous deflection history u(t) occurring at a specified depth and at a specified radial distance from the center of the applied load.

Green Function Solution

As discussed in the previous section, the general problem of dynamic response to transient loads is made tractable by resolving the temporal loads into a series of harmonic loads through the use of integral transforms. The problem is then reduced to that of determining the steady-state displacements that result from a unit harmonic load. That problem falls under the general category of wave propagation.

Though solutions for the propagation of waves in a homogeneous elastic halfspace date back to Lamb (1904), and those for homogeneous elastic continua date back even further, Thomson (1950) was the first to develop a theoretical formulation for the propagation of waves through a *layered* elastic halfspace. His approach was to develop a transfer matrix that linearly related the stresses and displacements at the top of a given material layer to the stresses and displacements at the bottom of the same layer. By enforcing continuity of the particle velocities and stresses at the layer interfaces, the stresses and displacements at the bottom of each layer could be equated to the stresses and displacements at the top of the next layer. In this way, the stresses and displacements at any layer interface could be computed from the stresses and displacements at the ground surface (which is just the top of the uppermost layer) by successively applying the matrix equation for each intervening layer.

Thomson's original derivation was for plane body waves. Haskell (1953) extended Thomson's matrix approach to the calculation of phase velocity dispersion curves for plane Rayleigh waves propagating along the surface of a layered medium. In the process, he corrected an error in Thomson's original derivation. This matrix approach to the analysis of wave propagation in layered elastic media has since come to be known as the Haskell-Thomson transfer matrix method.

Kausel and Roesset (1981) derived a complementary matrix formulation based on stiffness matrices rather than transfer matrices. Their stiffness matrices, which relate the stresses at the top and bottom of a given layer to the displacements at the top and bottom of the same layer, are analogous to the stiffness matrices commonly used in matrix structural analysis and finite element analysis. For media with multiple material layers, the stresses and displacements at the bottom of one layer are equated to the stresses and displacements at the top of the next layer and a global stiffness matrix is constructed by overlapping the individual layer stiffness matrices. The global stiffness matrix relates the stresses applied at all of the layer interfaces to the resulting displacements at all of the interfaces.

Both of the approaches outlined above for plane waves (line loads) can be extended to axisymmetric problems such as vertical point and disk loads by applying the principles of Fourier superposition to decompose the stresses and

displacements in the radial and azimuthal directions[8]. The resulting Green function solutions (e.g., Bouchon, 1981) are expressed in terms of integrals of sums of transcendental functions. These can only be evaluated numerically, which makes them computationally intensive and extremely inefficient.

Building on a concept developed by Lysmer and Waas (1972) for the finite element analysis of shear wave propagation, Kausel (1981) showed that by making the layer thicknesses small (for example, by discretizing the material layers), the transcendental functions that describe the displacements in the vertical direction (the direction of layering) could be replaced by a piecewise linear approximation. The resulting algebraic expressions could then be integrated in closed-form. Strictly speaking, the technique is only applicable to layered soils over a rigid half-space; however, it can be extended to the case of layered soils over an elastic half-space by combining an exact solution for the elastic half-space with the discretized solution for the overlying soil layers (Hull and Kausel, 1985).

The mathematical derivation of the elastodynamic Green functions will not be presented here. Instead, the reader is referred to Kausel and Peek (1982), who derive closed-form Green function solutions for a number of

---

[8] Just as temporal functions can be transformed into the frequency domain, spatial functions can be transformed into the wavenumber domain. The wavenumber, denoted by k, is the spatial equivalent of the circular frequency ω.

important loading cases such as disk, ring, and point loads. The solution procedure for the vertical displacements resulting from a vertical disk load applied on the ground surface is briefly summarized in the paragraphs that follow so that the reader has some idea of the computational steps involved.

The first step in the solution procedure is to subdivide the material layers (a task that will be discussed in more detail in the next section) and compute the sublayer stiffness matrices. The sublayer stiffness matrices are given by the quadratic function

$$\mathbf{K}_m = k^2\mathbf{A}_m + k\mathbf{B}_m + \mathbf{G}_m - \omega^2\mathbf{M}_m \tag{49}$$

where k is the wave number, ω is the circular frequency, and the matrices $\mathbf{A}_m$, $\mathbf{B}_m$, $\mathbf{G}_m$, and $\mathbf{M}_m$ (Table 6) are functions of the mass density (ρ), shear modulus (G), Lame's constant (λ), and thickness (h) of the material sublayer.

For linear elastic materials, the shear modulus is a real-valued constant. For viscoelastic materials that exhibit hysteretic damping, the real-valued shear modulus can be replaced by the complex modulus G*=G(1+i2D), where D is the damping ratio. Even though the damping ratio actually varies with the amplitude of the shear strain, it must be assumed constant here in order to maintain the linearity of the solution; otherwise, Fourier superposition could not be applied.

Table 6. Terms in the stiffness matrix equation
(after Kausel and Peek, 1982)

$$\mathbf{A} = \frac{h}{6}\begin{bmatrix} 2(\lambda+2G) & 0 & 0 & \lambda+2G & 0 & 0 \\ 0 & 2G & 0 & 0 & G & 0 \\ 0 & 0 & 2G & 0 & 0 & G \\ \lambda+2G & 0 & 0 & 2(\lambda+2G) & 0 & 0 \\ 0 & G & 0 & 0 & 2G & 0 \\ 0 & 0 & G & 0 & 0 & 2G \end{bmatrix}$$

$$\mathbf{B} = \frac{1}{2}\begin{bmatrix} 0 & 0 & \lambda-G & 0 & 0 & -(\lambda+G) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda-G & 0 & 0 & \lambda+G & 0 & 0 \\ 0 & 0 & \lambda+G & 0 & 0 & -(\lambda-G) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -(\lambda+G) & 0 & 0 & -(\lambda-G) & 0 & 0 \end{bmatrix}$$

$$\mathbf{G} = \frac{1}{h}\begin{bmatrix} G & 0 & 0 & -G & 0 & 0 \\ 0 & G & 0 & 0 & -G & 0 \\ 0 & 0 & \lambda+2G & 0 & 0 & -(\lambda+2G) \\ -G & 0 & 0 & G & 0 & 0 \\ 0 & -G & 0 & 0 & G & 0 \\ 0 & 0 & -(\lambda+2G) & 0 & 0 & \lambda+2G \end{bmatrix}$$

$$\mathbf{M} = \rho\frac{h}{6}\begin{bmatrix} 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \end{bmatrix}$$

ρ = mass density

G = shear modulus

λ = Lame's constant

h = layer thickness

The global stiffness matrix, **K**, is assembled next by overlapping the individual sublayer stiffness matrices, $\mathbf{K}_m$, at their interfaces in much the same way as for a finite element solution (Figure 39). The global stiffness matrix relates the loads applied at the layer interfaces ($\overline{\mathbf{P}}$) to the resulting particle displacements ($\overline{\mathbf{U}}$) at the interfaces:

$$\overline{\mathbf{P}} = \mathbf{K}\overline{\mathbf{U}} \tag{50}$$

The displacements can therefore be found by inverting the stiffness matrix:

$$\overline{\mathbf{U}} = \mathbf{K}^{-1}\overline{\mathbf{P}} = \mathbf{F}\overline{\mathbf{P}} \tag{51}$$

The inverted stiffness matrix is called the flexibility matrix and is denoted by **F**.

The stiffness matrix can be inverted using spectral decomposition. This necessitates determining the natural modes of wave propagation by solving the eigenvalue problem that results from setting the applied loads equal to zero:

$$\mathbf{K}\phi = \emptyset \tag{52}$$

The solution of this quadratic eigenvalue problem would ordinarily yield 6N eigenvalues, where N is the number of material sublayers. However, half of the propagation modes represented by those eigenvalues correspond to waves

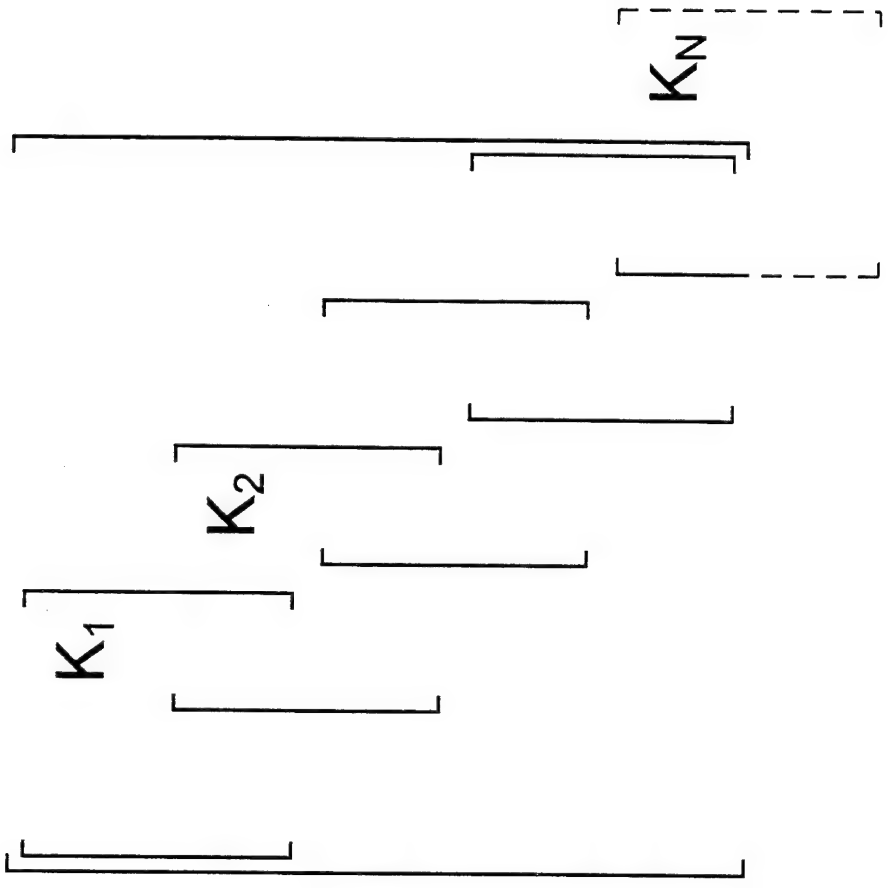


Figure 39. Assembling the global stiffness matrix

propagating toward the source rather than away from it and can therefore be ignored. The remaining 3N eigenvalues can be further separated into those corresponding to anti-plane motion (termed Love modes) and those corresponding to in-plane motion (termed Rayleigh modes). Being uncoupled, they can be obtained by solving two separate linear eigenvalue problems. The first involves only those stiffness matrix terms corresponding to tangential motion; the second involves only the stiffness matrix terms corresponding to vertical and radial motion. For the case of a vertical disk load, only the Rayleigh modes are excited; therefore, only the second eigenvalue problem needs to be solved. The solution of that reduced eigenvalue problem produces 2N eigenvalues, k_n, with corresponding eigenvectors, ϕ_n. The eigenvalues are the wavenumbers of the natural propagation modes and the eigenvectors are the corresponding radial and vertical displacements at the layer interfaces.

The individual eigenvalues and eigenvectors are combined to construct the desired flexibility matrix. The terms of the flexibility matrix pertaining to vertical displacements resulting from vertical excitations can be expressed as

$$f^{ij} = \sum_{n=1}^{2N} \frac{\bar{\phi}_n^i \, \bar{\phi}_n^j}{k^2 - k_n^2} \tag{53}$$

where $\bar{\phi}_n$ denotes the vertical displacement components of eigenvector ϕ_n and the superscripts j and i are used to designate the matrix and vector

elements corresponding to the layer interfaces where the load is applied and the displacement is to be calculated, respectively.

Once the flexibility matrix has been constructed, the final step is to apply the appropriate loading conditions and calculate the resulting displacements. For the case of a uniform vertical disk load with a radius of R applied at the ground surface, the elements of the load vector $\overline{\mathbf{P}}$ are zero for all of the layer interfaces except the one corresponding to the ground surface. At the ground surface, the horizontal component of the load is zero and the vertical component is given by

$$P(r,\theta) = \begin{cases} 1 & \text{for} \quad 0 \le r \le R \\ 0 & \text{for} \quad r > R \end{cases} \tag{54}$$

Using the principle of Fourier superposition, this can be transformed into the wavenumber domain using Fourier transforms in the circumferential direction (θ) and Hankel transforms in the radial direction (r). Because the uniform disk load is independent of the direction angle, θ, the Fourier transform is zero for all terms except the first (n=0) and the transformed load reduces to

$$p(k,n=0) = -\int_0^\infty J_0(kr)\, r\, dr = -\frac{R}{k} J_1(kR) \tag{55}$$

Here J_0 and J_1 denote Bessel functions of the first kind and orders zero and one, respectively.

The transformed load vector $\bar{\mathbf{p}}$ is multiplied by the flexibility matrix to determine the displacements in the wavenumber domain. Inverse Fourier and Hankel transforms are subsequently applied to transform the displacements back into the spatial domain. The transformed solution in the spatial domain is given by

$$u^{ij}(r) = \begin{cases} R\sum_{n=1}^{2N} \frac{\bar{\phi}_n^i \bar{\phi}_n^j}{k_n}\left[\frac{\pi}{2ik_n} J_0(k_n r) H_1^{(2)}(k_n r) - \frac{1}{Rk_n^2}\right] & \text{for} \quad 0 \le r \le R \\ R\sum_{n=1}^{2N} \frac{\bar{\phi}_n^i \bar{\phi}_n^j}{k_n}\left[\frac{\pi}{2ik_n} J_1(k_n r) H_0^{(2)}(k_n r)\right] & \text{for} \quad r \ge R \end{cases} \tag{56}$$

where $H_0^{(2)}$ and $H_1^{(2)}$ denote Hankel functions of the second kind and orders zero and one, respectively. Note that $j = 1$ will be always be used here since the desired Green function solutions are for a surface load. Similarly, $i = 1$ should be used to obtain the surface deflections; however, values other than one can also be used to obtain the vertical displacements at different depths. The displacements occurring at depths that do not correspond to layer interfaces can be found by linear interpolation between the appropriate interface values.

To summarize, the procedure used to solve for the Green functions consists of the following steps:

1. discretize the pavement system
2. compute the layer stiffness matrices (Equation 49)
3. construct the global stiffness matrix (Figure 39)
4. solve the eigenvalue problem (Equation 52)
5. compute the displacements (Equation 56)

Green functions evaluated at the radial offsets of each FWD sensor are multiplied by the frequency-domain components of the transformed FWD loading pulse (Equation 45) to obtain the frequency-domain components of the resulting surface deflections. The deflection histories recorded by each sensor can then be recovered by applying the inverse Fourier transform given in Equation 48. Finally, the peak values of these displacement pulses are combined to form the FWD deflection basin.

Discretizing the Pavement System

The Green function solution above is based upon the linearization of the transcendental functions describing displacement as a function of depth. This is achieved by subdividing each of the material layers into sublayers. Obviously, the thinner the sublayers, the closer the discrete solution will come to matching the exact solution. Unfortunately, the increased accuracy afforded by a finer discretization comes at the expense of increased computation time.

The majority of that time is spent in solving the eigenvalue problem—a task that varies as $O(N^2)$. As a result, a small increase in the number of sublayers can result in a considerable increase in computation time. Because thousands of deflection basins had to be generated to train the neural network, it was crucial that the computation time for each pavement profile be minimized. This meant using the thickest sublayers possible and discretizing to the shallowest depth possible without adversely affecting the accuracy of the solution.

Based on a parametric study involving an anti-plane line load applied at the ground surface (chosen because a closed-form solution was available for comparison), Sanchez-Salinero (1987) recommended that:

1. the ratio of the sublayer thickness to the wavelength of the shear waves in the material layer being discretized should be somewhere between 1:6 and 1:8
2. the sublayer thickness near the ground surface should be no greater than half the radial distance between the source and the point where the surface displacements are measured
3. the depth of the transmitting boundary should be no less than half the radial distance between the source and the point where the surface displacements are being measured
4. the depth of the transmitting boundary should be no less than four times the wavelength of the shear waves in the fastest material layer.

The last two of these recommendations reflect the fact that the transmitting boundary provides only a gross approximation of a halfspace, so it must be placed sufficiently deep so that it does not adversely affect the solution.

In addition to the heuristics enumerated above, Roesset[9] recommends that the sublayer thickness close to the ground surface be no greater than half the radius of the disk load. Although the definition of "close" is subjective, the application of St. Venant's principle suggests that this rule be applied to a depth of at least four times the radius (i.e., twice the diameter) of the disk load. The combined thickness of the surface and base layers varies from less than one to more than seven times the radius of the disk load. For simplicity, and making sure to be somewhat conservative, a maximum sublayer thickness of 0.25 ft was specified everywhere above a depth of 4 ft (eight times the radius of the disk). This heuristic therefore applies throughout the depth of the surface and base layers and into the subgrade material.

Extensive experimentation with many different layer thicknesses and layer properties revealed that there is also a minimum sublayer thickness of approximately 0.125 ft. Whenever thinner sublayers were used, the eigensolver became unstable and the stiffness matrix could not be inverted successfully. In addition, it was found that the surface layer had to be subdivided into at least

---

[9] Personal communication.

two, and preferably three, sublayers. When fewer layers were used, the surface deflections were inexact for several of the pavement profiles with large velocity contrasts between the surface and base layers. Similarly, it was found that a minimum of two or three sublayers was preferable in the base layer as well because the curvature of the deflection function in that area can be very severe (e.g., see Figure 40). If the base were to be treated as a single layer, the peak deflection could be missed entirely, resulting in a significant underestimation of the surface deflections.

In summary, the surface and base layers had to be subdivided into as few sublayers as possible without exceeding the maximum thickness limit of 0.25 ft while being careful to have at least two, and preferably three, sublayers in each material layer. This latter rule was waived for the thinnest surface layers so the sublayers would not become too thin and violate the minimum thickness limit of 0.15 ft.

In the subgrade, the heuristics that govern both the sublayer thickness and the maximum depth depend on the wavelength of shear waves, and therefore on frequency. To simultaneously accommodate both the thin sublayers needed at the higher frequencies and the great depths needed at the lower frequencies would mean building an extremely large stiffness matrix that would require considerable computational resources to invert. Because the stiffness matrix is a function of frequency, the eigenvalue problem must be

solved all over again for each different frequency. There is therefore no advantage to using the same discretization for all frequencies. Instead, it makes sense to discretize the subgrade anew to take advantage of the changing wavelengths. Because the criteria for the sublayer thickness and the maximum depth are both linear functions of wavelength, they can be satisfied simultaneously by using the same minimal number of sublayers at each different frequency and simply increasing their thickness in proportion to the increasing wavelengths.

The recommendation made by Sanchez-Salinero that the transmitting boundary be placed at a depth at least four times the wavelength of the fastest material in the profile was developed by studying soil profiles that became stiffer with depth. In a pavement system, the profile generally becomes softer with depth. If the same heuristic were applied here, the combination of the high wave speeds in the asphalt layer and the low frequencies produced by the FWD would require that the subgrade be discretized to excessive depths. For example, at a frequency of 1 Hz, the wavelength of a shear wave propagating through asphalt with a wave speed of 5000 ft/s is nearly a mile! Even if the requirement was modified to be four times the wavelength in the deepest material, a typical wave speed of 500 ft/s in the subgrade would require a discretization depth of close to half a mile. Instead, an alternative criteria for discretization depth had to be developed.

According to Miller and Pursey (1955), approximately 2/3 of the energy imparted by a vertical disk load on the surface of a uniform elastic halfspace is dissipated as Rayleigh waves. Furthermore, according to Ewing, Jardetzky, and Press (1957), Rayleigh waves attenuate much more slowly with distance than the body waves that dissipate the remainder of the energy. The Rayleigh waves attenuate as $1/r^{0.5}$ whereas the body waves attenuate as $1/r^2$ at the ground surface. A suitable criteria for locating the transmitting boundary can therefore be developed by examining the propagation depth of the Rayleigh waves, since they predominate.

For a Rayleigh wave of wavelength λ propagating across the surface of an uniform elastic halfspace, the vertical particle motions are given by

$$u(z) = \frac{2\bar{q}}{\bar{s}^2 + 1} \exp(-2\pi\bar{s}\frac{z}{\lambda}) - \bar{q}\exp(-2\pi\bar{q}\frac{z}{\lambda}) \qquad (57)$$

where

$$\bar{q} = \sqrt{1 - \frac{v_R^2}{v_P^2}}$$

$$\bar{s} = \sqrt{1 - \frac{v_R^2}{v_S^2}}$$

and v_P, v_S, and v_R are the wave speeds of the compression, shear, and Rayleigh waves, respectively. The wave speed ratios are functions of the Poisson's ratio of the material:

$$\frac{v_R^2}{v_S^2} = K^2$$

and

$$\frac{v_R^2}{v_P^2} = \alpha^2 K^2$$

where

$$\alpha^2 = \frac{1-2\nu}{2-2\nu}$$

and

$$K^6 - 8K^4 + (24 - 16\alpha^2)K^2 + 16(\alpha^2 - 1) = 0$$

This last equation can be solved as a cubic equation in K^2. For a Poisson's ratio of 0.35, $\alpha^2 = 0.2308$, and the solution of the cubic equation is $K^2 = 0.8743$. For that value of K^2, the equation for vertical particle motions becomes

$$u(z) = 1.3191 \exp(-2.2281\frac{z}{\lambda}) - 0.8934 \exp(-5.6137\frac{z}{\lambda}) \quad (58)$$

which can be expressed in a dimensionless form by scaling the displacements at depth by the displacement at the ground surface:

$$\overline{u}(z) = \frac{u(z)}{u(0)} = 3.099 \exp(-2.2281\frac{z}{\lambda}) - 2.099 \exp(-5.6137\frac{z}{\lambda}) \quad (59)$$

This equation is plotted in Figure 40 to a depth of three wavelengths. Clearly, the vast majority of the particle motion occurs in the upper three wavelengths of the halfspace. At a depth of three wavelengths, the vertical displacement is only 0.4 percent of the displacement at the ground surface. Even at a depth of two wavelengths, the vertical displacements is only 3.6 percent of the displacement at the ground surface. Therefore, from the standpoint of determining the vertical deflections of the ground surface, there is no need to place the transmitting boundary at a depth greater than roughly twice the Rayleigh wavelength.

Several parametric studies were conducted to validate this conclusion. The first showed that the deflection basins calculated using maximum depths of three wavelengths were no different from those calculated using a maximum depth equal to two wavelengths. Another showed that the deflection basins calculated with a reflecting (rigid) boundary placed at a depth of two

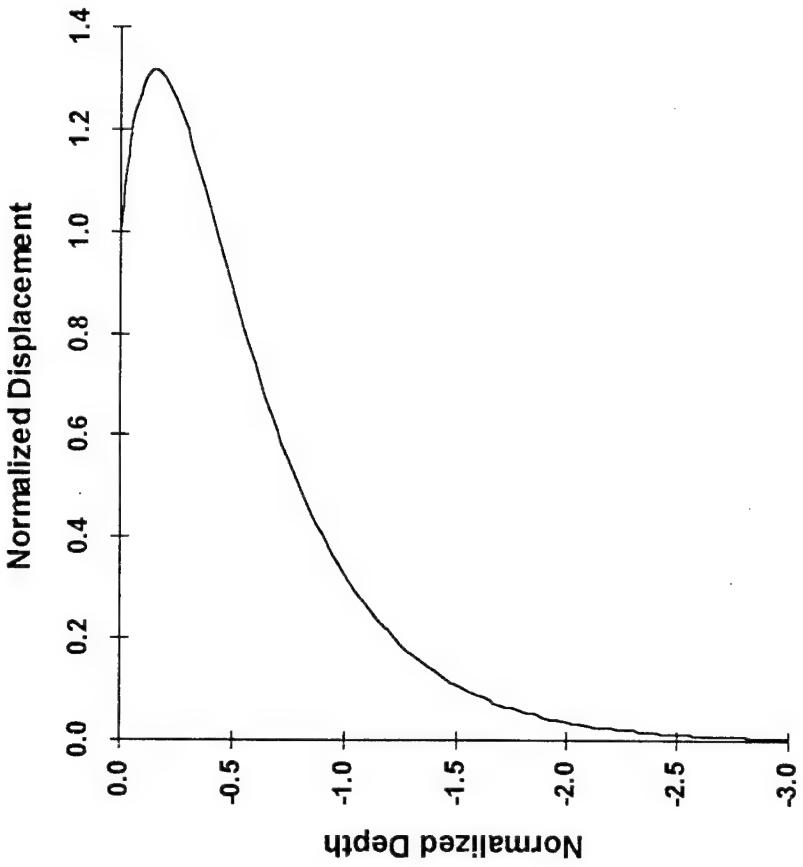


Figure 40. Vertical displacements induced by a Rayleigh wave

wavelengths were no different from those calculated with a transmitting boundary placed at the same depth. This, too, suggests that a maximum depth of two wavelengths is sufficient.

It should not be inferred from these results that there were no differences in the deflection *histories*, only that there were no differences in the deflection *basins* because any errant body waves produced by either the inexact halfspace approximation or the rigid bedrock arrived back at the surface too late to influence the peak deflections.

Rayleigh waves propagating in a layered medium are dispersive: their propagation velocity is a function of the depth to which they propagate. Unfortunately, this means that their wavelengths (calculated as propagation velocity divided by frequency) are a function of the depths to which they propagate, which, in turn, are a function of their wavelengths. This implicit relationship between wavelength and material properties suggests an iterative solution. The following convergent trial-and-error method was adopted:

1. using the Rayleigh wave speed in the subgrade, establish a wavelength corresponding to the frequency of interest
2. compute an adjusted wave speed using a weighted average of the wave speeds in each material layer
3. using this adjusted wave speed, establish a new wavelength and repeat until convergence is obtained.

The weighting factors used to compute the weighted average wave speed can be obtained from the relative contributions of each material layer to the total energy dissipated by the Rayleigh wave. These contributions can be approximated by integrating Equation 59 to obtain the area under the particle velocity curve:

$$\int_0^z \bar{u}(x)\,dx = 1.017 - 1.3909\exp(-2.2281\frac{z}{\lambda}) + 0.3739\exp(-5.6137\frac{z}{\lambda}) \qquad (60)$$

By evaluating this integral at each layer interface and subtracting one result from the next, the relative contributions of each layer can be determined.

To illustrate this procedure, assume that the pavement profile has the properties shown in Figure 41 and the frequency of interest is 50 Hz. Assume, also, that Poisson's ratio is 0.35 in all of the layers so that Equation 59 can be used to compute the particle motions as a function of depth. Based on a wavespeed of 500 ft/s in the subgrade, the initial estimate for the Rayleigh wavelength is 10 ft. Substituting that wavelength and the 0.5-ft thickness of the surface layer into Equation 60, the area under the particle motion curve within the surface layer is 0.055. Using the 2.0-ft depth to the bottom of the base layer, the area under the curve from the ground surface to the bottom of the base layer is 0.248, so the area under the curve within the base layer is 0.193. Since the areas under the curve must add up to 1.017 (which is found by

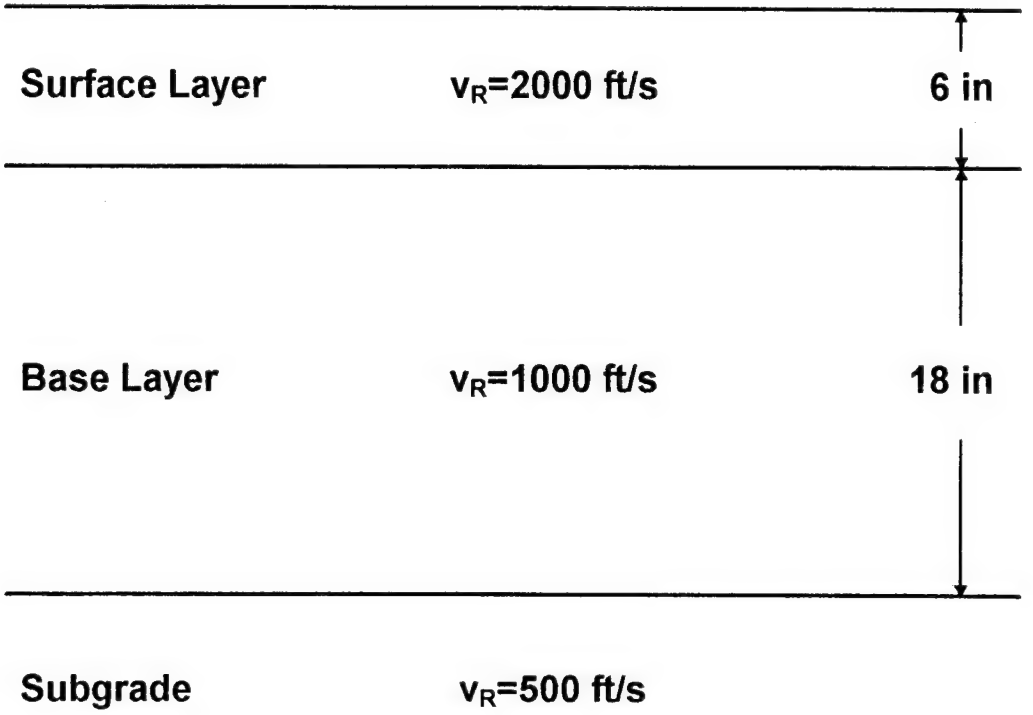


Figure 41. Pavement profile used for Rayleigh wavelength calculation

evaluating Equation 60 at $z = \infty$), the area under the curve within the subgrade material is 0.769. The adjusted wavespeed is therefore

$$v_s = \frac{0.055}{1.017}(2000) + \frac{0.193}{1.017}(1000) + \frac{0.769}{1.017}(500) = 676 \text{ fps}$$

and the adjusted wavelength is

$$\lambda = \frac{676 \text{ fps}}{50 \text{ s}^{-1}} = 13.52 \text{ ft}$$

The analysis is now repeated using this new wavelength to normalize the particle motion curve. After three more iterations, the solution will have converged to an answer of 12.7 ft. By calculating the actual Green function solutions for this profile and measuring the radial distances between surface points having the same phase angle, this can be shown to be a very good approximation of the Rayleigh wavelength in this profile.

Using the procedure above to determine wavelength, the subgrade was discretized for each individual frequency to a maximum depth of two Rayleigh wavelengths. The total number of sublayers in the subgrade was allowed to vary depending on the frequency. At high frequencies, fewer sublayers were needed because the relatively short wavelengths meant that the majority of the particle motion occurred in the surface and base layers. As a result, the degree

of curvature in the particle motion function was relatively low in the subgrade, so it could be adequately linearized with relatively few sublayers. On the other hand, at the lowest frequencies, the maximum deflection occurs well within the subgrade; therefore, relatively more sublayers were required in the subgrade in order to maximize the resolution of the particle motions over depth.

Modeling the FWD Load History

The dynamic load imparted to the pavement by the FWD is generated by a free-falling mass impacting a steel plate. A rubber pad beneath the plate uniformly distributes the load to the pavement, and a series of rubber buffers above the plate decelerate the falling mass and condition the loading pulse. A typical loading pulse has a duration of 25–30 msec and a magnitude that varies as the square root of the drop height. As mentioned in Chapter 2, the loading pulse is meant to approximate the deflection pulse created by a moving truck wheel. Based on the work by Barksdale (1971), the typical pulse duration of 25–30 msec would correspond to a vehicle speed of approximately 35 mph.

For programming convenience and computational flexibility, a functional analogue to the FWD loading pulse that could be easily varied in both amplitude and duration was desired. Researchers at the University of Texas (e.g., Foinquinos, Roesset, and Stokoe, 1993a; Kang, 1990; Chang, et. al., 1992) use a triangular approximation to the loading pulse. Lukanen (1992) suggests that the ideal loading pulse is a haversine:

$$p(t) = \frac{P}{2}\left(1 - \cos\frac{2\pi t}{T}\right) \qquad (61)$$

Here P is the peak amplitude and T is the pulse duration. That shape, shown in Figure 8, is qualitatively the same as the theoretical stress pulses produced by Barksdale (Figure 42). Both the triangle and the haversine were investigated for use in generating the synthetic deflection basins.

Lukanen (1992) presents a series of measured loading pulses corresponding to different drop heights, pavement types, and buffer configurations. The results obtained using the standard buffers on a flexible pavement are shown in Figure 43. The four loading pulses (indicated by light lines) were normalized to a unit load and averaged to arrive at a "typical" FWD loading pulse (indicated by the heavy line). That average measured pulse is compared to triangle and haversine approximations in both the time and the frequency domains in Figures 44 and 45, respectively. Both functional analogues were made to have the same duration (26.5 msec) and peak amplitude (1.0) as the average measured pulse.

Figure 45 shows that the frequency-domain components for all three pulses are remarkably similar (especially at frequencies below 75 Hz, where all three curves essentially drop to zero), despite the fact that neither of the functional analogues captures the double peak exhibited by the measured pulse in the time domain (Figure 44). Since all of the calculations in the Fourier

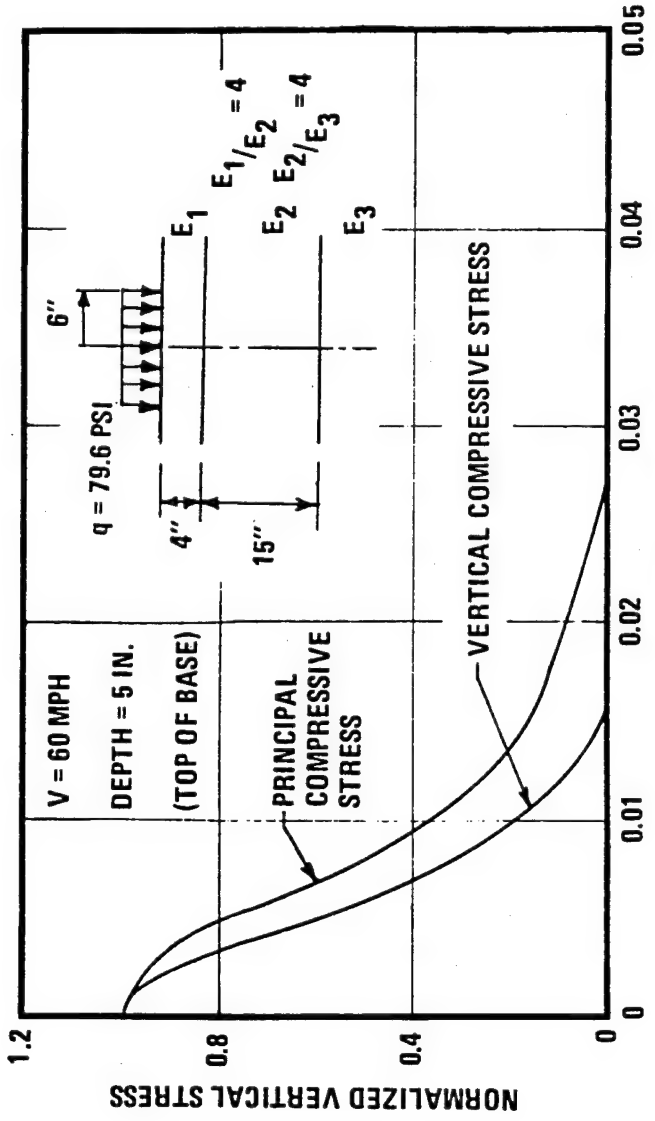


Figure 42. Stresses applied by a moving wheel (Barksdale, 1971)

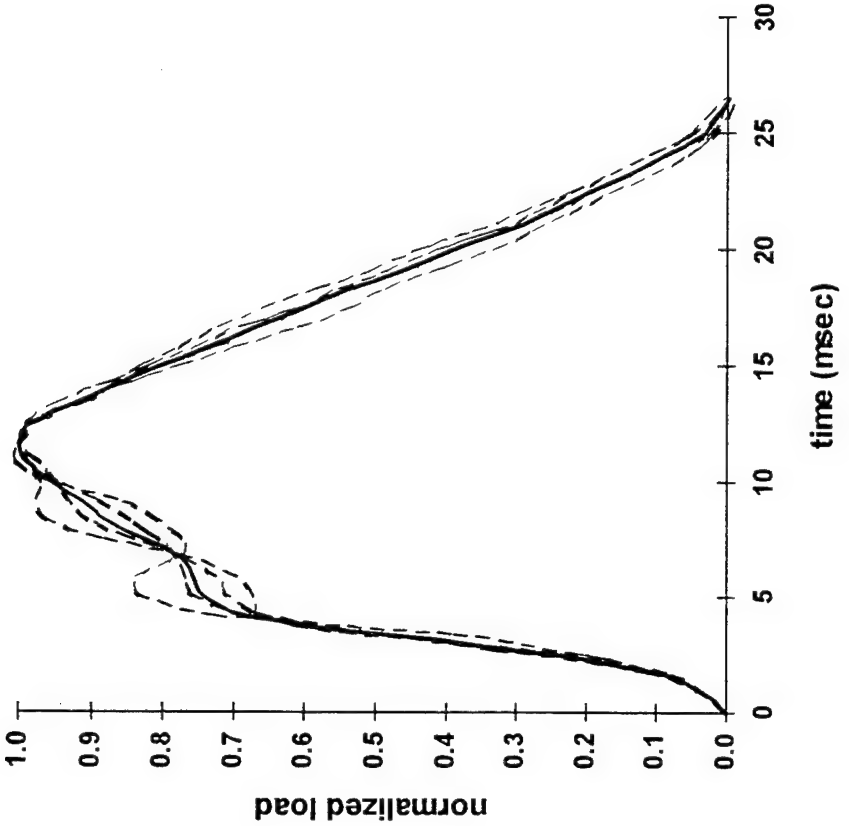


Figure 43. Measured and average FWD load pulses (after Lukanen, 1993)

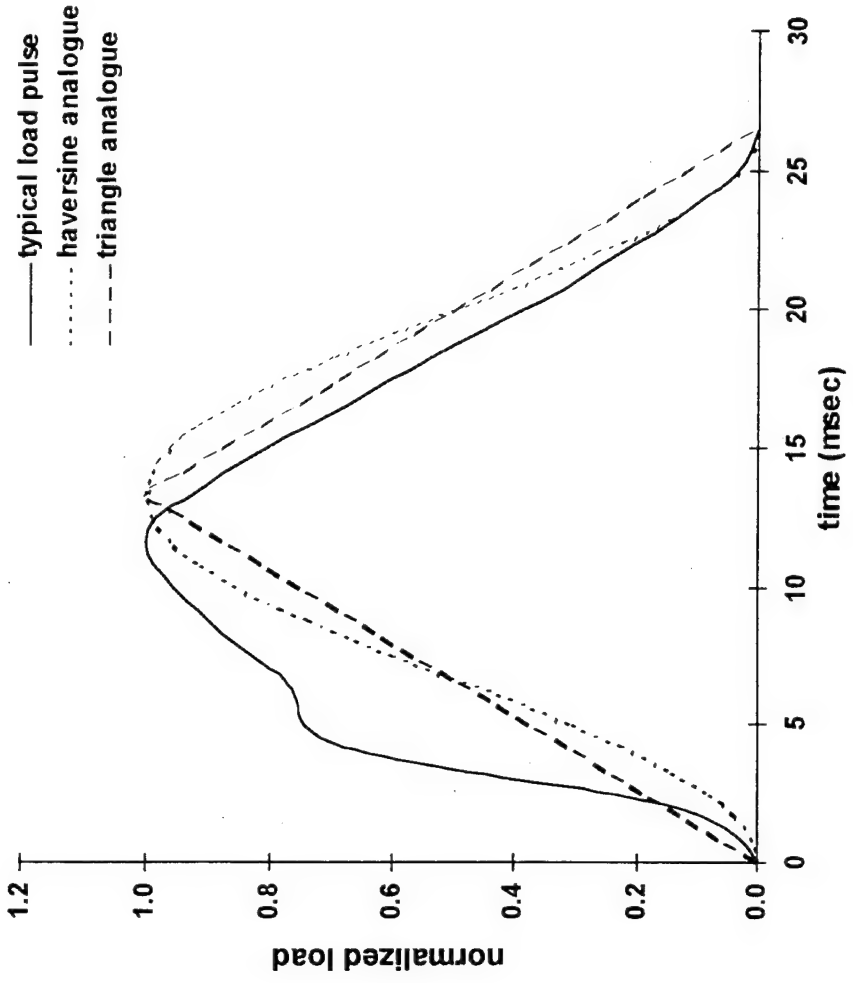


Figure 44. Average measured load pulse and functional analogues in the time domain

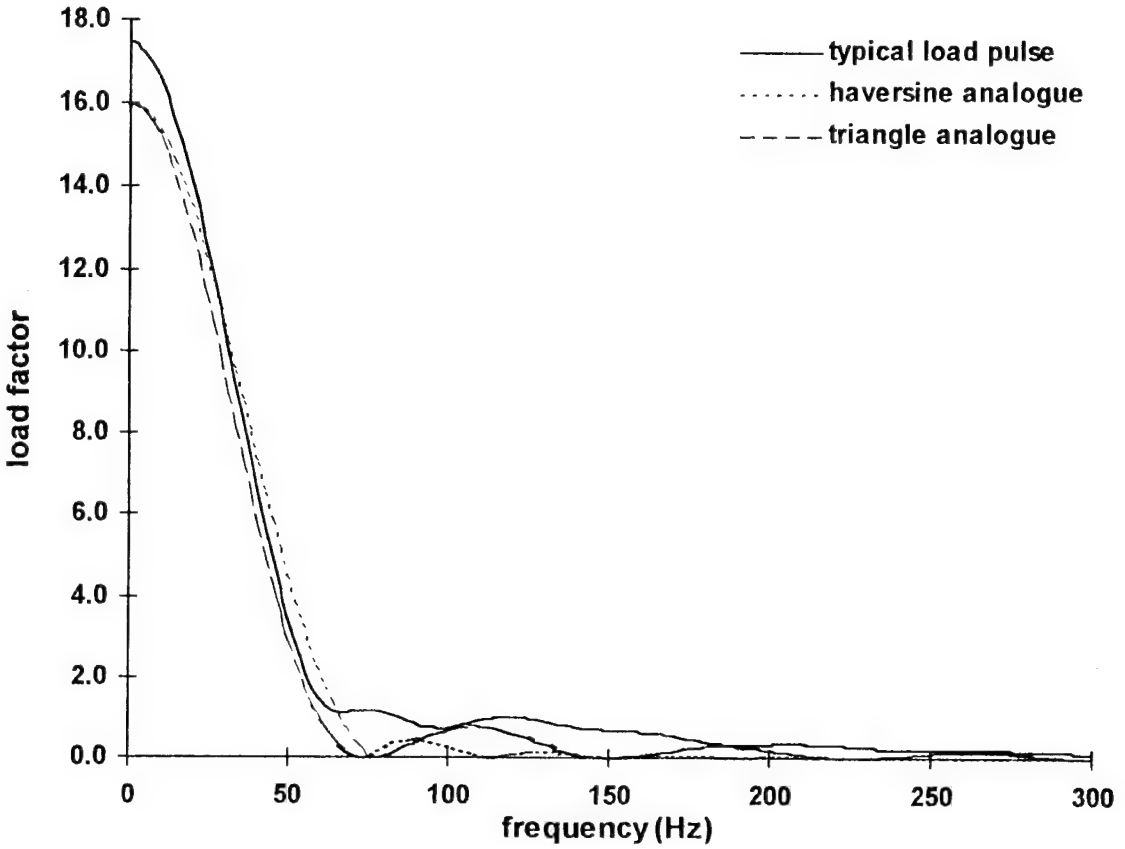


Figure 45. Average measured load pulse and functional analogues in the frequency domain

superposition analysis occur in the frequency domain, both analogues appear to be suitable surrogates for the measured pulse.

Both of the functional analogues have slightly lower DC offsets (zero frequency intercepts) because the total impulse—the area under the force-time curve—is slightly less. The total impulse for the average measured pulse is 17.4 msec and the total impulse for the functional analogues is 16.0 msec. The peak amplitudes of the analogues would therefore have to be increased by approximately nine percent (17.4/16.0=1.09) to obtain the same total impulse without changing the pulse durations.

The haversine was chosen over the triangle for this study because it better approximates (at least aesthetically) the shape of an "ideal" load pulse and is slightly better behaved in the frequency domain. The perfectly straight sides of the triangular pulse and the sharp discontinuity at its peak result in spurious higher-frequency components that do not exist in the haversine or the measured pulse. Figure 46 compares the measured FWD pulse with an impulse-adjusted haversine analogue given by

$$p(t) = 0.545\left(1 - \cos\frac{2\pi t}{26.5}\right) \qquad (62)$$

Again, despite the apparent differences in the time domain, there is very close agreement in the frequency domain over the range 0–75 Hz. At frequencies

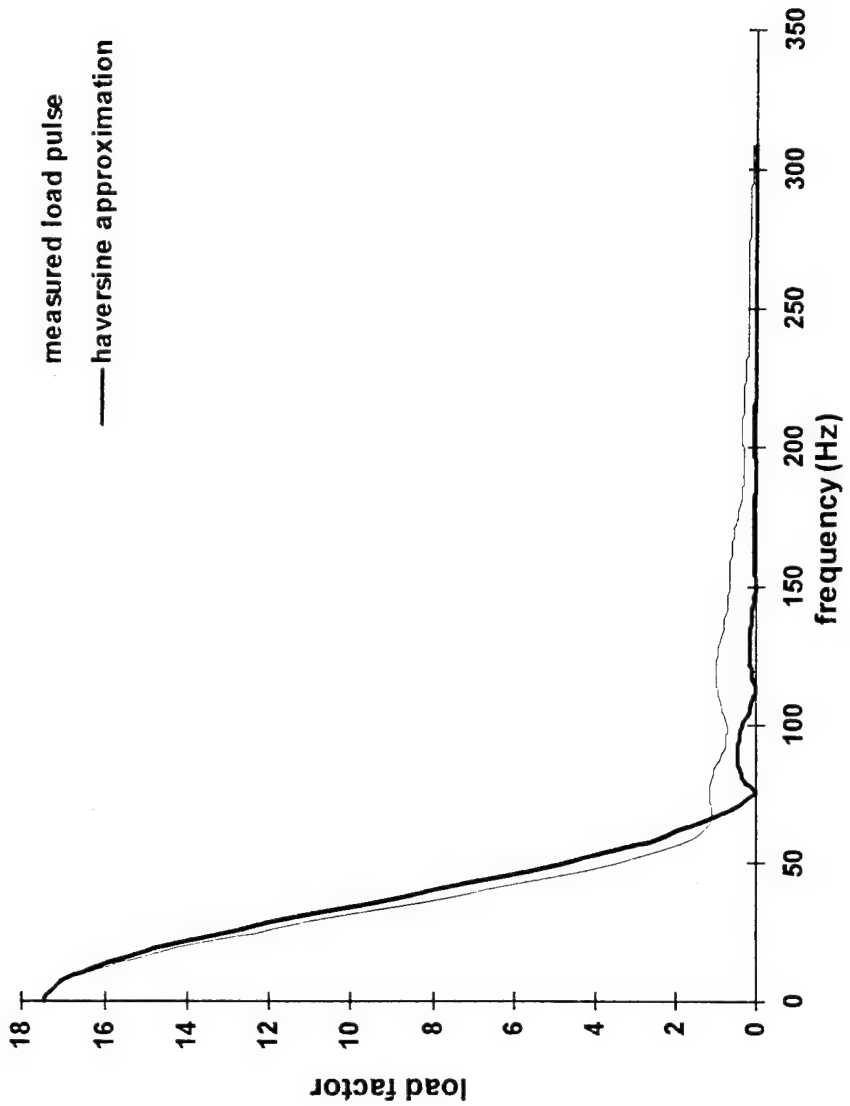


Figure 46. Average measured load pulse and adjusted haversine analogue

above 75 Hz, the magnitudes of both pulses remain near zero anyway, so any differences at those frequencies are immaterial.

At a frequency of $0.5T^{-1}$ = 75.47 Hz, the FFT of the haversine pulse has a magnitude identically equal to zero. It would be convenient to use this as a frequency cutoff to limit the bandwidth that must be considered in the Fourier superposition analysis. Because the computational costs incurred in the analysis vary in direct proportion to the number of frequencies that must be analyzed, it was important to either minimize the total bandwidth or maximize the widths of the frequency intervals.

In order to show the feasibility of using a bandwidth-limited analysis, a 512-point FFT was performed on a 32-point haversine analogue calculated using Equation 62. Next, all of the components at frequencies higher than 75.47 Hz were set to zero. A 512-point inverse FFT was then performed to recover the time-domain loading pulse. The solid line in Figure 47 represents the original haversine and the symbols show the loading pulse recovered from the bandwidth-limited frequency components. This clearly shows that almost nothing is lost by limiting the bandwidth.

The 32-point bandwidth-limited haversine shown in Figure 47 was therefore accepted as the functional analogue of the FWD loading pulse. The Fourier superposition analysis would therefore require that Green function solutions be obtained for 31 discrete frequencies spaced at 2.36-Hz intervals.

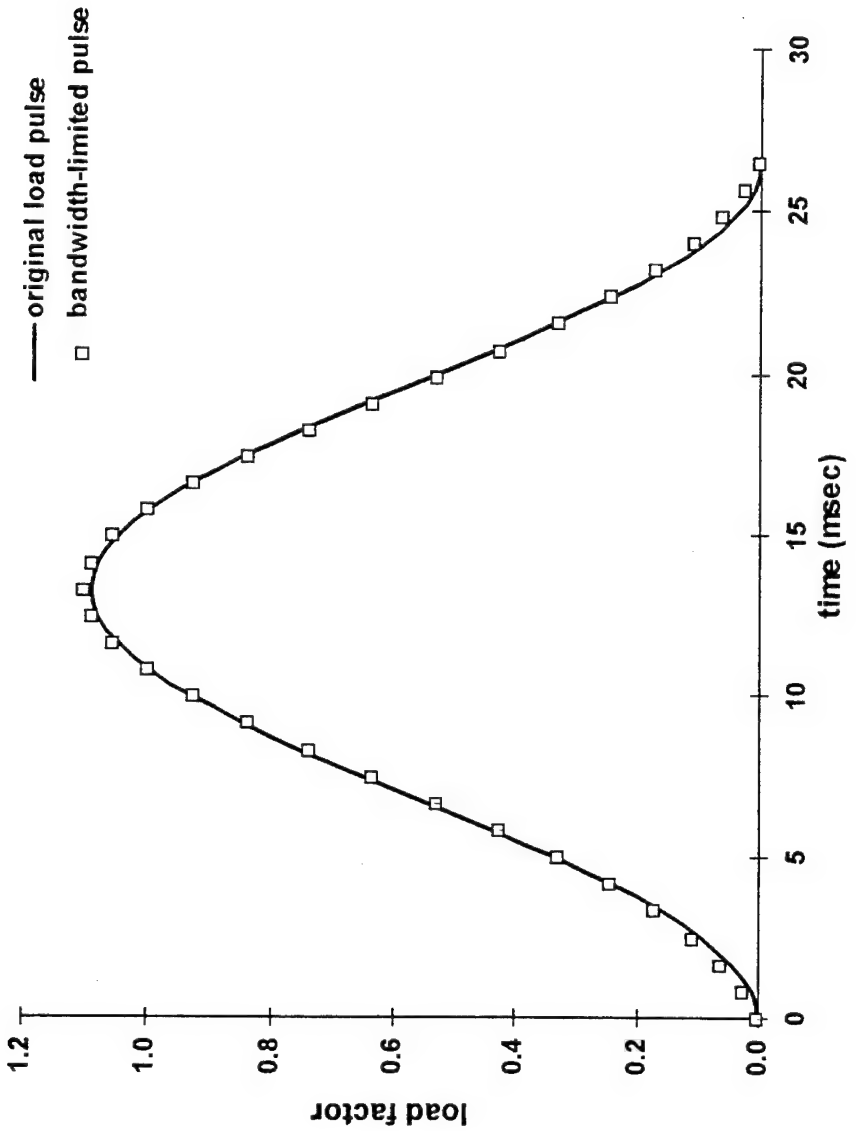


Figure 47. Original and bandwidth-limited haversine analogues

(Because the FFT of the haversine load pulse is zero at the 32nd point, which corresponds to the frequency cutoff of 75.47 Hz, there was no need to develop a solution at that frequency.)

Training Set Generation

Kausel implemented his Green function solutions in a FORTRAN computer program called PUNCH. Another FORTRAN implementation was developed by Sanchez-Salinero (1987) while studying under Kausel's collaborator, Roesset, at the University of Texas at Austin. The latter program was chosen to compute the synthetic deflection basins needed in this phase of the research. The program was optimized for deflection basin generation by removing all coding that did not directly support the calculation of vertical deflections resulting from a vertical disk load (e.g., the solution of the eigenvalue problem for the Love modes). Simultaneously, coding was added to implement the frequency-dependent discretization of the pavement layers and to compute the discrete Fourier transforms. The latter was accomplished using the Fast Fourier Transform (FFT) algorithm developed by Cooley and Tukey (1965). The specific implementation is described in Press, et. al. (1989).

The modified FORTRAN code was verified using examples from Kang (1990) and Foinquinos, Roesset, and Stokoe (1993a). A check was also made to ensure that the static solution produced by the program (i.e., for a frequency at or very close to zero) coincided with the solutions obtained using WESLEA.

The results were, indeed, identical within the finite accuracy of the discrete solutions.

Once verified, the FORTRAN program was used to generate deflection basins for all of the 10,000 pavement profiles created in Phase I. Despite all of the effort that went into optimizing the Green function code and minimizing the number of material layers, the program still used approximately 5 seconds of processing time (on the Cray Y-MP) to compute each deflection basin. As a result, it took approximately 10 hours to generate the entire training set. That is roughly 20 times longer than it took to generate the static training set using WESLEA. This disparity clearly illustrates the disadvantages of using dynamic pavement analyses in conventional basin-matching programs. Based on the fact that WESDEF needed nearly 40 minutes to perform a static backcalculation on 250 deflection basins, a similar task using a dynamic analysis would take more than 13 hours!

Summary

This chapter has described the creation of the neural network training set for the second phase of the research. The synthetic deflection basins were calculated using an elastodynamic analysis of pavement deflections instead of the static analysis used previously. The retraining of the neural network and the examination of its backcalculation results are presented in the next chapter.

CHAPTER 6

NEURAL NETWORK TRAINING AND TESTING

This chapter describes the training of several artificial neural networks using deflection basins generated by the static, layered-elastic analysis program WESLEA. The first two networks were trained to backcalculate pavement layer moduli directly from the "perfect" deflection basins output by WESLEA. Those networks differ only in the sensor spacings used to define the deflection basins. A third network was trained with "noisy" deflection basins to make it more robust in the presence of real-world experimental data. The speed and accuracy of that "robust" network are subsequently examined relative to the conventional basin-matching programs WESDEF (which uses the iterative approach) and MODULUS (which uses a database approach). The "robust" network was subsequently retrained during the second phase of the research using deflection basins generated by the elastodynamic Green function solutions for the pavement's response to FWD loads. The accuracy of that network is also examined. All of the neural network training described here was accomplished on the Cray Y-MP supercomputer at WES using the computer code listed in Appendix A.

Neural Network Training with Perfect Basins

As mentioned in Chapter 4, there is no single standard for positioning the geophones in an FWD test. The positioning used most often with the Dynatest device is a uniform 1-ft spacing. The specifications for SHRP's LTPP project include a non-uniform spacing that concentrates more deflection sensors close to the load. This is done in order to better define the moduli in the upper layers of the pavement system. Because it would be extremely difficult to design one neural network that could accommodate either spacing, two separate networks were trained, instead. One network used deflection basins defined by a uniform 1-ft sensor spacing. The other network was based on the SHRP offsets of 0, 8, 12, 18, 24, 36, and 60 in. Because the database of training examples had been built using all of the most commonly used sensor offsets, training the two networks was simply a matter of extracting the appropriate seven deflections from the database.

Determining the Network Architecture

When training conventional backpropagation networks, the network architecture must be established before the start of training. The first step in establishing the architecture is to determine the number of processing layers in the network. At a minimum, the network must have two layers: the input layer and the output layer. Multi-layer, feed-forward neural networks generally have one or more hidden layers as well.

In principle, a network consisting of just one hidden layer can be taught to approximate any continuous functional mapping (Cybenko, 1989; Hornik, Stinchcombe, and White, 1989). In practice, however, multiple hidden layers often allow the same functional mapping to be learned faster (Freeman and Skapura, 1991) and with fewer neurons (Hecht-Nielson, 1990). Prior experience has shown that mappings of the form $\Re^m \rightarrow \Re^n$ (i.e., from one real space to another) are often better learned by networks with two hidden layers. Initial experimentation with neural networks having a single hidden layer bore this out. Therefore, three-layer networks (i.e., networks with two hidden layers and an output layer) were used for all subsequent training[10].

The second step in establishing the network architecture is to determine the number of neurons in each layer. The number of neurons in the input and output layers is easy to determine—they are simply the number of input and output parameters, respectively. Unfortunately, there are no well-established procedures for choosing the number of neurons in each of the hidden layers. The optimum architecture must strike a balance between insufficient knowledge

---

[10] Early in the history of neurocomputing, a network with an input layer, an output layer, and one hidden layer was termed a "three-layer" neural network. The same neural network today is usually called a "two-layer" neural network in recognition of the fact that the input layer does nothing more than distribute the network inputs to the neurons in the first hidden layer. Because no processing is done in the input layer, it is no longer counted.

capacity (too few connections) and excessive capacity (too many connections). If the network has insufficient capacity, it will be incapable of accurately performing the required mapping. On the other hand, if the network has excessive capacity, it may simply "memorize" all of the training examples. In that case, the network will be incapable of performing mappings for deflection basins that were not included in the training sets. In other words, it will not *generalize* well. For the initial phase of this research, simple trial-and-error was used to select an appropriate number of neurons in the hidden layers.

Training the Network

Network training began by scaling all of the input and output data to a range appropriate for the neural network. Because the sigmoidal logistic function (Figure 48) has an output range of [0,1], all of the target outputs must be scaled to fit within that same range; otherwise, the network outputs will never match the target outputs and the network will never learn. Similarly, the useful input range for the sigmoidal logistic function is approximately [−6,+6] so the *weighted sums* of the network inputs should fall somewhere within that range. If the magnitudes of the weighted sums are too large, they will be indistinguishable from one another (they will all map to either 0 or 1) and the network will never learn.

If all of the network inputs are scaled to a range of [0,1] and the network weights are initialized to random numbers drawn from a uniform distribution

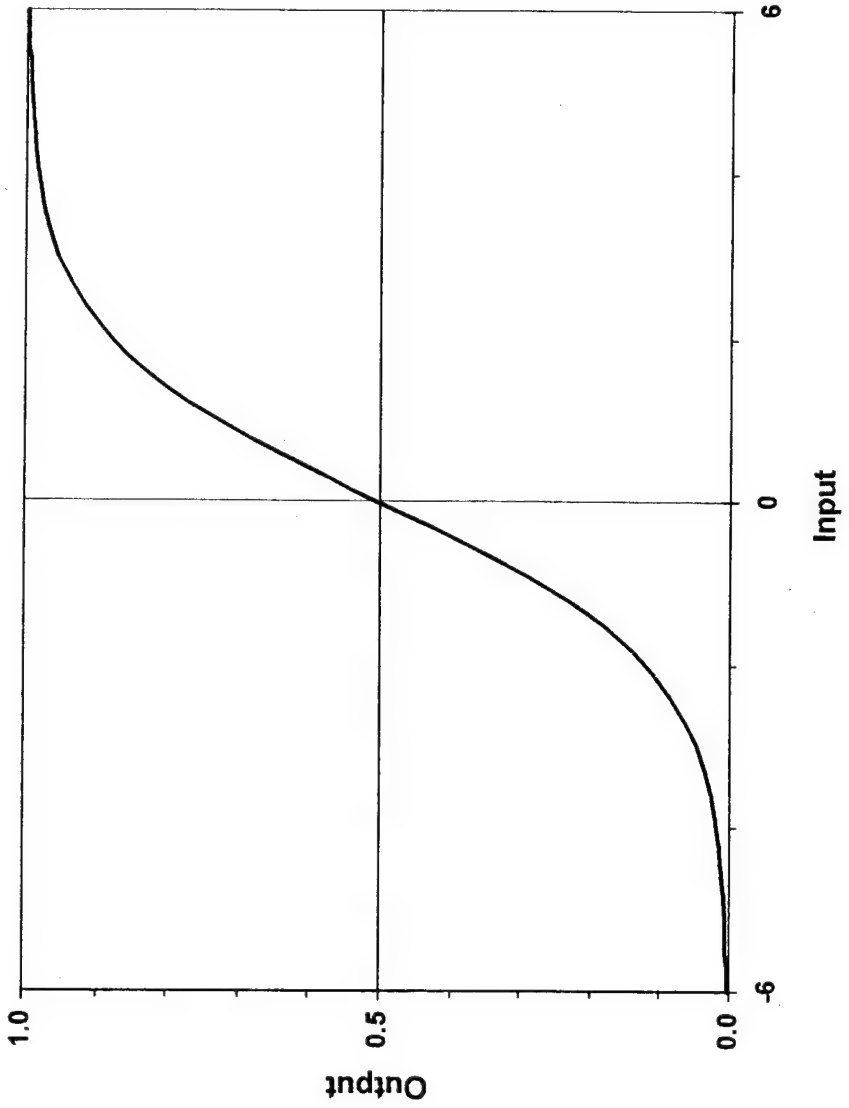


Figure 48. Effective ranges of the sigmoidal logistic function

covering the range [−0.1,+0.1], then all of the weighted sums calculated in the first hidden layer will fall between −0.9 and +0.9 (because there are nine inputs). This range covers the steepest, and therefore most sensitive, portion of the sigmoidal transfer function; therefore, the individual inputs will be clearly distinguishable from one another from the very start of training. In other words, valuable training time will not be wasted simply trying to determine the proper magnitude for the weights connecting each input to the hidden layer.

Because the pavement layer moduli and layer thicknesses were already uniformly distributed over specified ranges according to the design of the training set, it was a simple matter to rescale each of them to a [0,1] range using the values provided in Table 2. This was not true, however, for the deflection basins. All of the deflections had to be scaled by the same factor so as not to lose potential information about the *differences* between deflections at adjacent sensors. In other words, the entire deflection basin had to be scaled rather than the individual deflections.

Figure 49 shows a cumulative frequency distribution of all the deflections in the training database. Despite the fact that the maximum deflection is nearly 70 mils, the vast majority of the deflections (94 percent) fall between 0 and 10 mils. By simply dividing all of the deflections by 10, the majority were made to fall in the desired [0,1] range; furthermore, all but the largest one or two would fall comfortably within the useful range of [−6,+6].

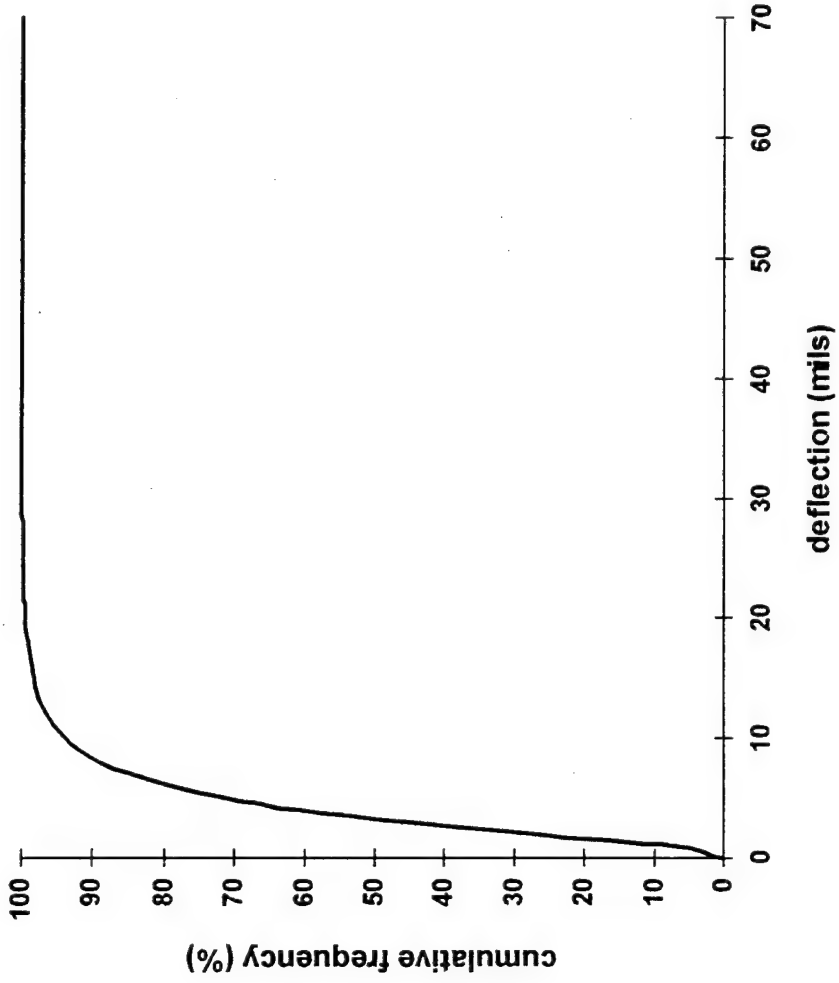


Figure 49. Cumulative frequency distribution of the deflections in the training set

Once the training set had been appropriately scaled, training proceeded by iteratively presenting the training examples to the network. Each pass through the set of 10,000 examples constituted a *training epoch*. During each training epoch, the first 9,750 examples in the training database were used to train the network. The remaining 250 examples were reserved for use as an independent test set. (It is always a good idea to test neural networks with an independent data set to make sure they are not simply memorizing the training set at the expense of being able to generalize to examples they have never seen before.)

Training progress was monitored by computing a mean squared error (MSE) for all of the training examples. The MSE was computed as

$$MSE = \frac{1}{mN}\sum_{k=1}^{N}\sum_{j=1}^{m}\left(y_j^k - t_j^k\right)^2 \qquad (63)$$

where m is the number of output neurons and N is the number of training examples. Figure 50 shows a typical training history. At first, the network's output error drops rapidly as training epochs are completed. With further training, the error asymptotically approaches some minimum level. At some point before that level is reached, the increase in accuracy that may accrue from any continued training is usually far outweighed by the cost of the additional training. At that point, training is halted.

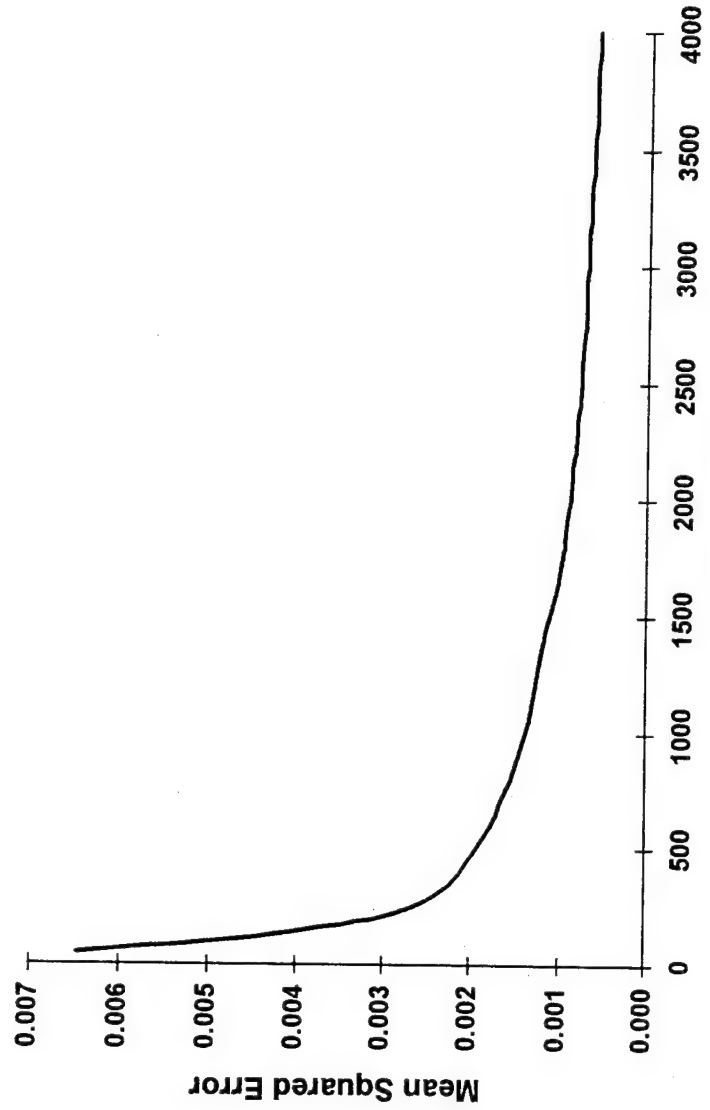


Figure 50. Typical network training history for a network trained with "perfect" deflection basins

In order to ensure that the network has reached the global minimum on the error surface rather than a local minimum, it should be retrained using weights initialized to different random values. Because the network retraining begins from a completely different location on the error surface, it is unlikely that the same local minima will be encountered. If the retrained network attains the same error level as before it is usually assumed that the global minimum has been reached. Most of the networks discussed here were trained two or more times, sometimes with different learning rates as well as different initial weights, to ensure they had, indeed, reached the global minimum.

After experimenting with several different architectures, the network architecture shown in Figure 51 was selected for having a low network output error relative to the number of hidden-layer neurons. The input layer of the network contains nine neurons which simply distribute the surface layer thickness (designated h_{ac} in the figure), the base layer thickness (designated h_b), and the seven deflections (designated as d_0 through d_6), respectively, to the first hidden layer. The first hidden layer contains eleven neurons and the second contains eight neurons. The output layer contains three neurons—one each for the surface, base, and subgrade moduli being backcalculated (designated as E_{ac}, E_b, and E_s, respectively). Two identical networks with this architecture were trained using deflection basins corresponding to both the SHRP sensor offsets and a uniform 1-ft spacing.

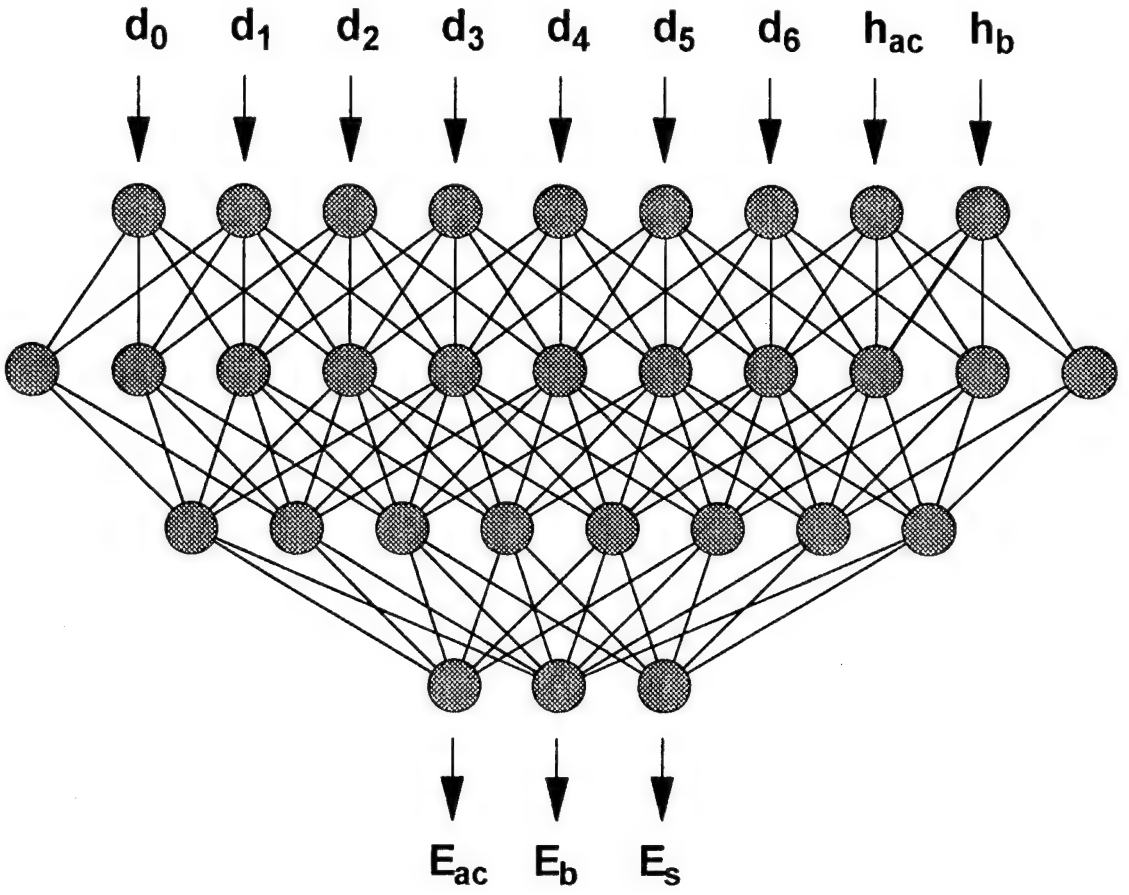


Figure 51. Network architecture used for backcalculating moduli from "perfect" deflection basins

Comparison Between Uniform and SHRP Sensor Spacings

As shown in Figure 52, the neural network trained with the SHRP sensor offsets achieved a lower mean squared error than the network trained with the uniform sensor spacing. Even more telling is the histogram in Figure 53 which compares the mean squared errors of the individual outputs at the conclusion of training. It is readily apparent that the network with the SHRP sensor spacings learned the mappings for the surface and base layer moduli more accurately while achieving the same accuracy in backcalculating the subgrade moduli. This shows very conclusively that repositioning sensors closer to the load provides better information from which to backcalculate the moduli in the upper layers of the pavement. Based on this finding, the SHRP sensor offsets were used for the remainder of the research.

Comparison Between Calculated and Target Moduli

Figures 54, 55, and 56 are scatter plots comparing the computed and target moduli for the asphalt, base, and subgrade layers, respectively. The plots show the results for all 250 of the deflection basins in the independent testing set. It is evident from these plots that the network successfully learned to backcalculate the pavement layer moduli from the synthetic deflection basins. In a broader context, these results are significant because they clearly show that neural networks can in fact be taught to solve complex, nonlinear inverse problems using training data generated by repeatedly solving the

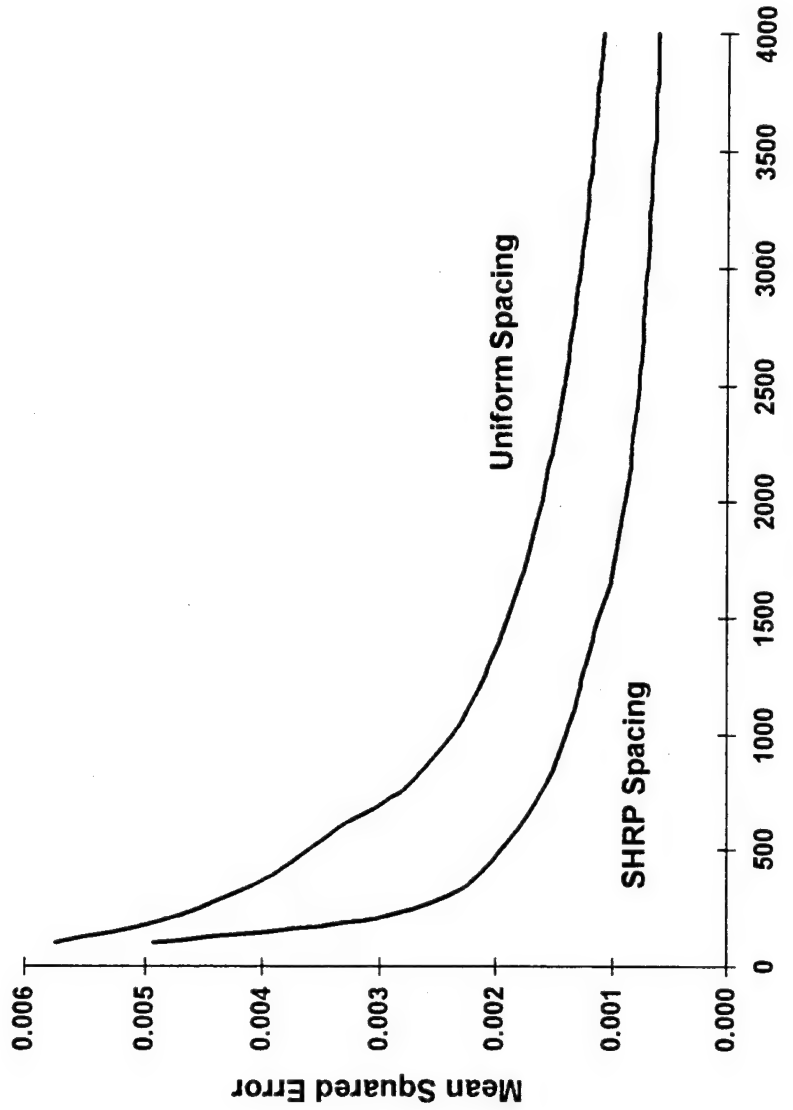


Figure 52. Network training histories using different sensor spacings

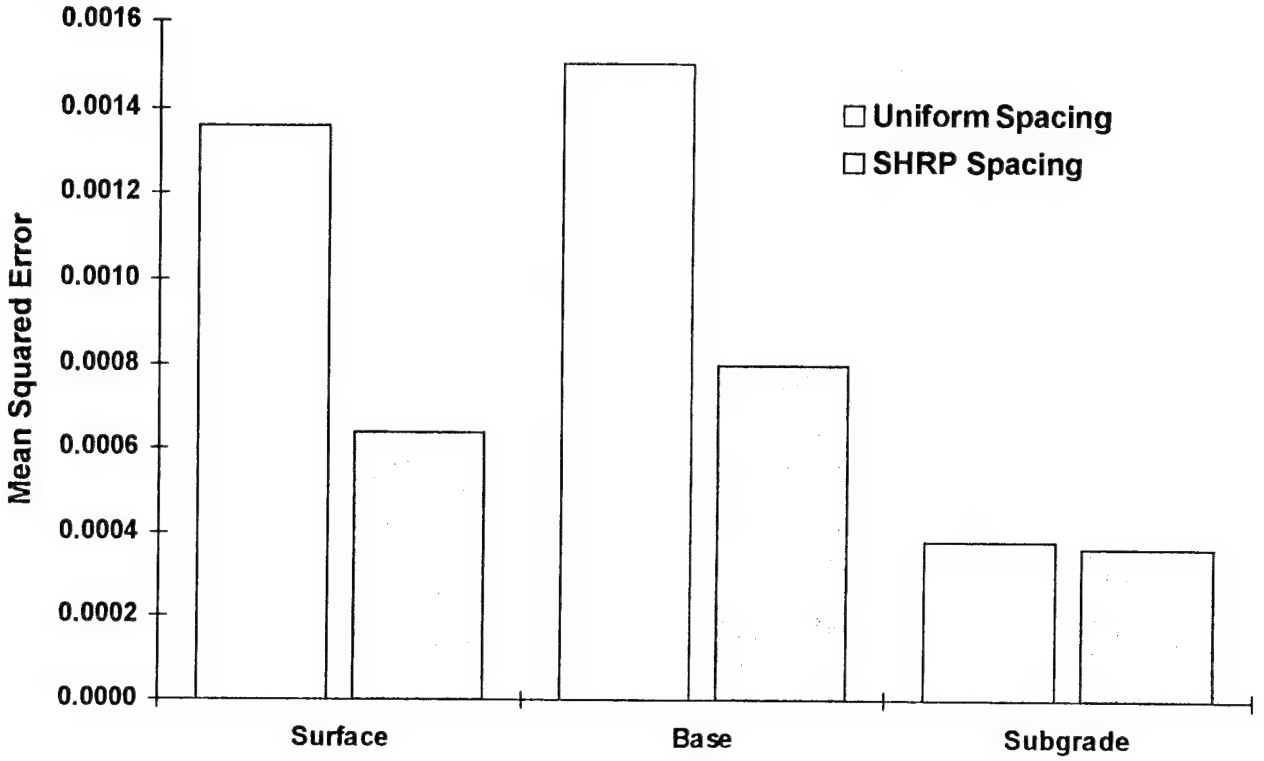


Figure 53. Comparison of network output errors after 4000 training epochs

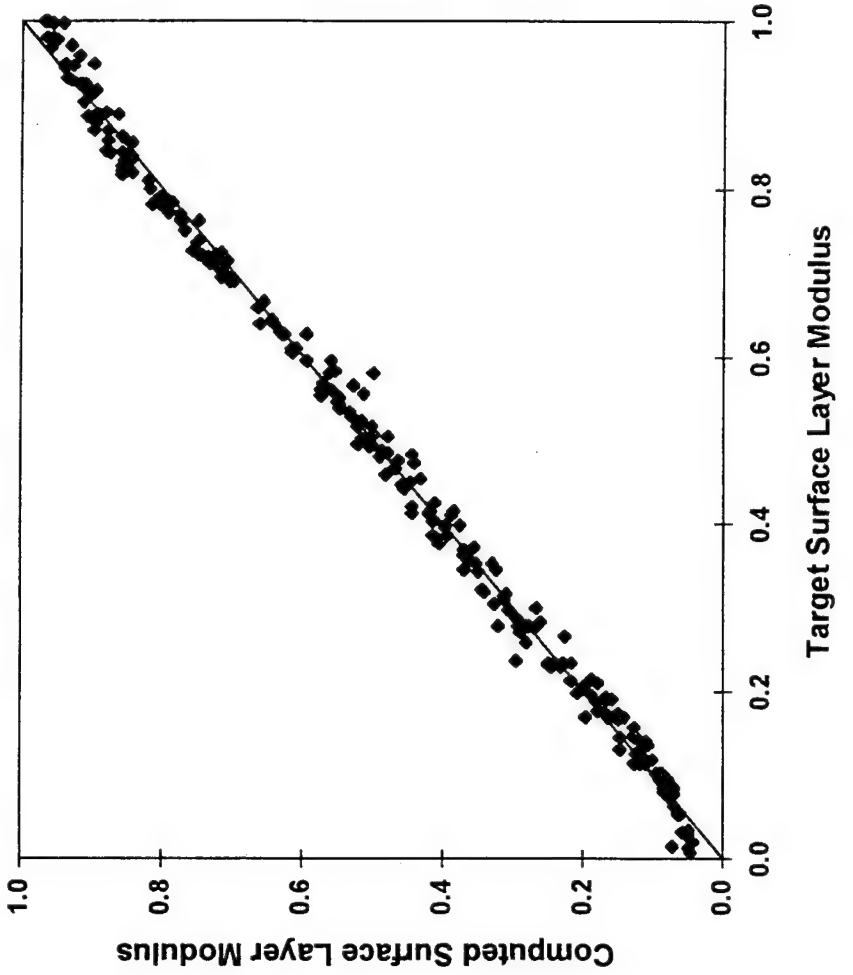


Figure 54. Normalized surface layer moduli from network trained using "perfect" deflection basins

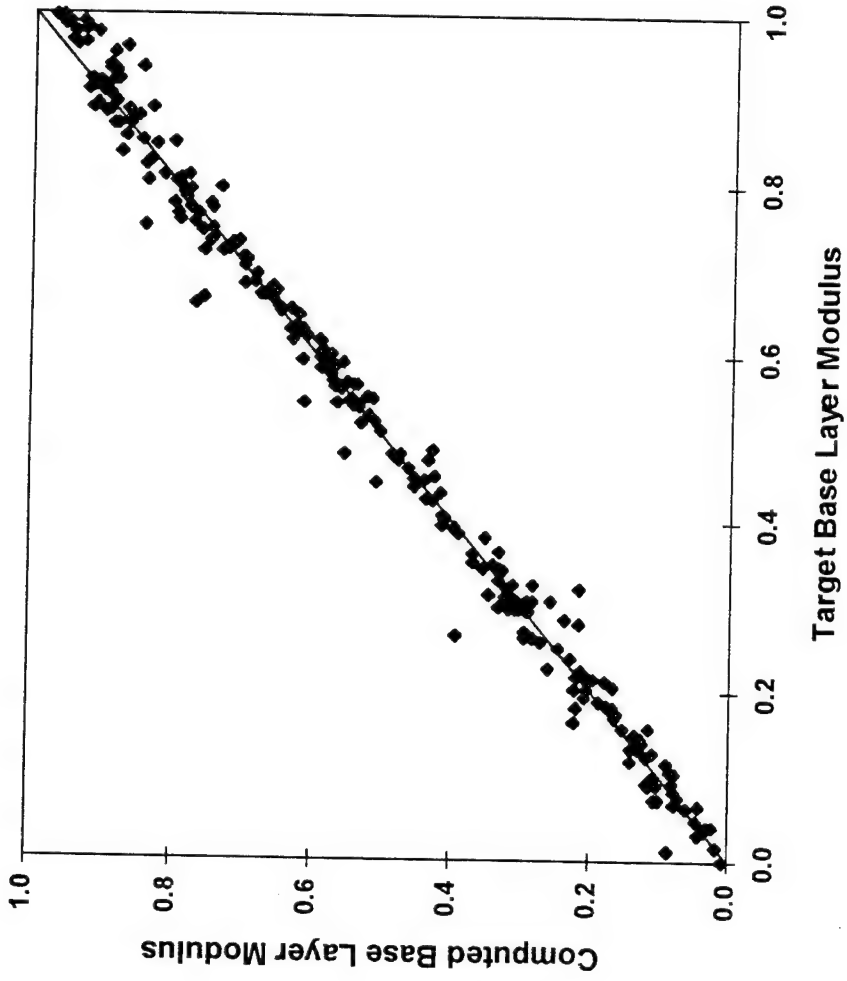


Figure 55. Normalized base layer moduli from network trained using "perfect" deflection basins

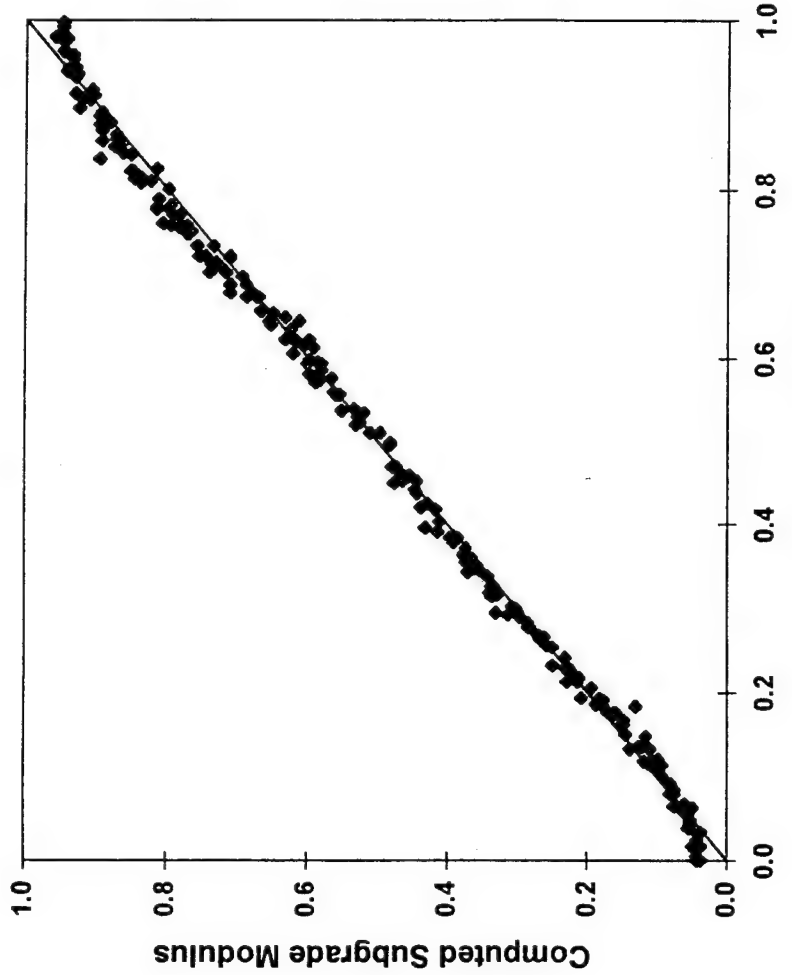


Figure 56. Normalized subgrade moduli from network trained using "perfect" deflection basins

forward problem. Together, the neural network and the forward problem solution form a closed loop (illustrated in Figure 19) in which half of the loop implements the forward problem while the other half (the neural network half) implements the inverse problem. This concept is not confined to geophysical data inversion; it can be applied to a wide variety of inversion problems in many different fields.

Neural Network Training with Imperfect Basins

Accurate deflection measurements are essential if the backcalculated layer moduli are to be correct. It would be unrealistic, however, to expect experimental deflections to exactly match the synthetic deflections used to train the neural networks. Even if the pavement system is well represented by the mathematical model used to calculate the synthetic deflection basins, real-world experimental data are subject to a variety of equipment-related measurement errors.

There are two primary types of equipment-related measurement error: systematic errors and random errors. The former are repeatable errors that can be minimized by proper calibration of the measurement apparatus; the latter are simply random measurement errors (noise) that cannot be reproduced. The specifications for the FWD test (ASTM, 1993) require that the systematic error for each geophone be no greater than 2% of the measured deflection and the repeatability error be no greater than 0.08 mils.

In order to determine the neural network's ability to handle deflection basins contaminated with noise, the 250 deflection basins in the independent test set were modified by adding random noise to each deflection. The random noise was drawn from a uniform distribution with limits of ±0.08 mils. These "noisy" deflection basins were then input to the neural network. Figures 57, 58, and 59 compare the computed and target moduli for the asphalt, base, and subgrade layers, respectively. It is obvious from these plots that a neural network trained with perfect deflection basins is not robust enough to accommodate noise.

One approach to developing a more robust network is to include random noise in the deflection basins used to train the network—a technique known as "noise injection" (Matsuoka, 1992). The introduction of random noise during training makes the network more robust because it has to learn to produce accurate moduli from inaccurate deflection basins. The final step in the first phase of the research was therefore to train a more robust version of the neural network by adding random noise to each of the seven deflections just before presenting them to the network. In order to accommodate both systematic errors and repeatability errors, the random noise was drawn from uniform distributions whose limits were equal to the larger of ±2% of the ideal deflection or ±0.1 mils. The latter was made slightly larger than the test specification to permit some room for error.

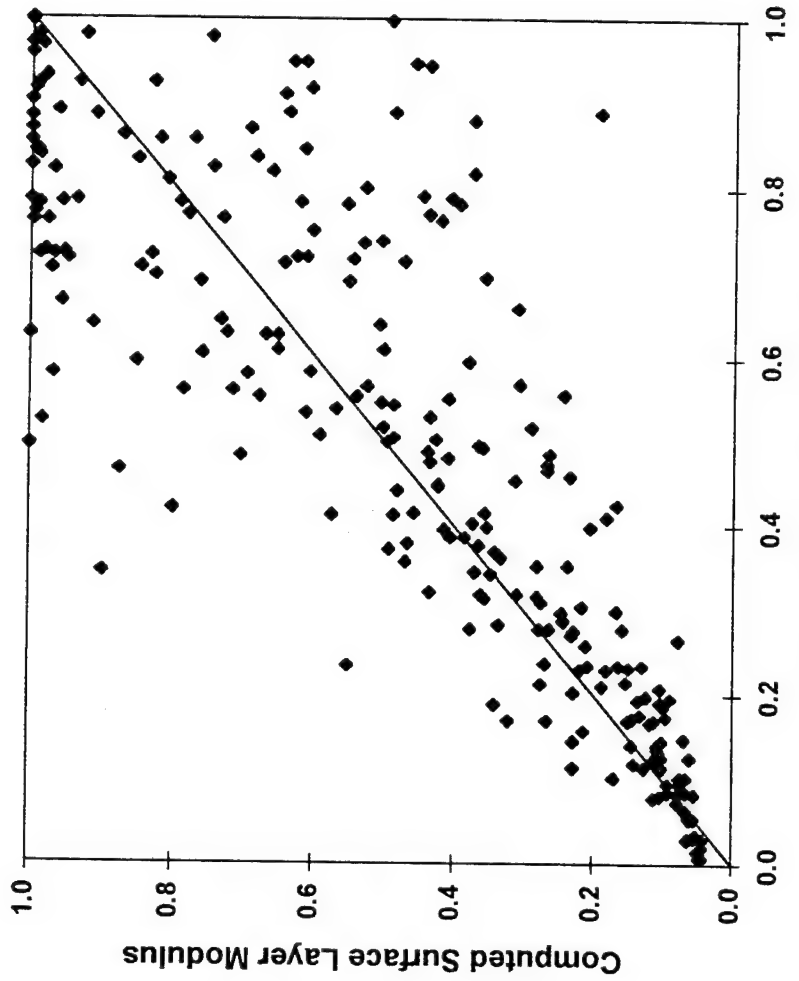


Figure 57. Normalized surface layer moduli backcalculated from "noisy" deflection basins

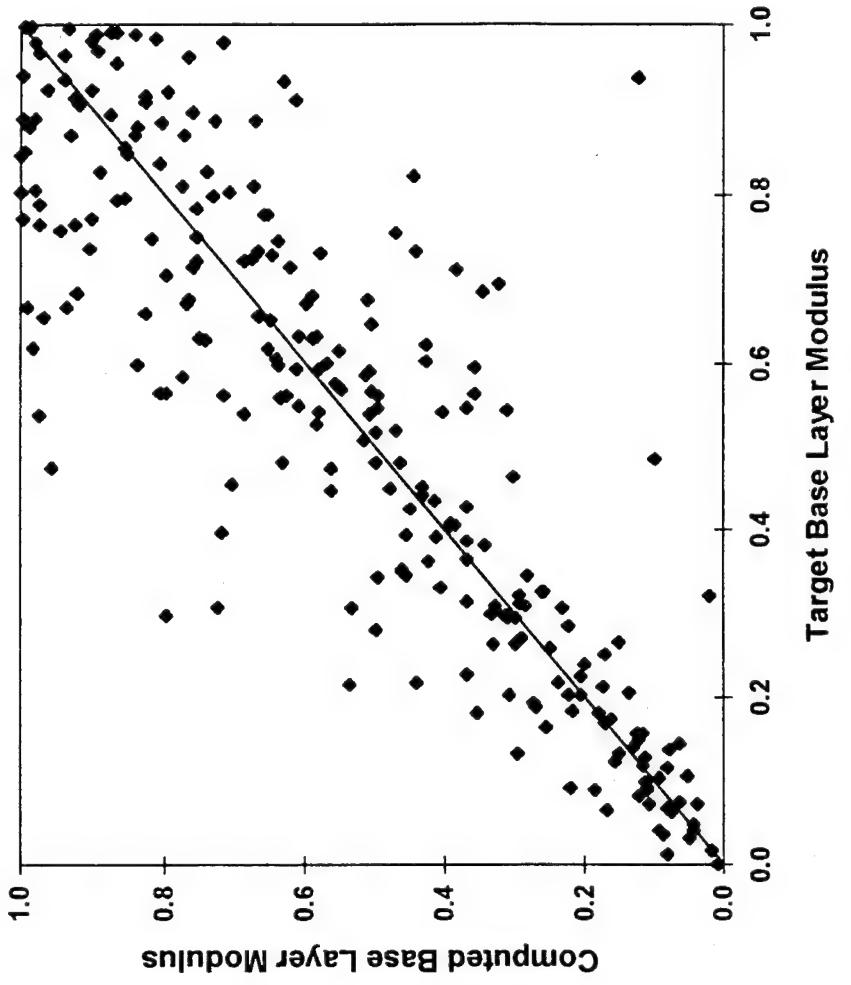


Figure 58. Normalized base layer moduli backcalculated from "noisy" deflection basins

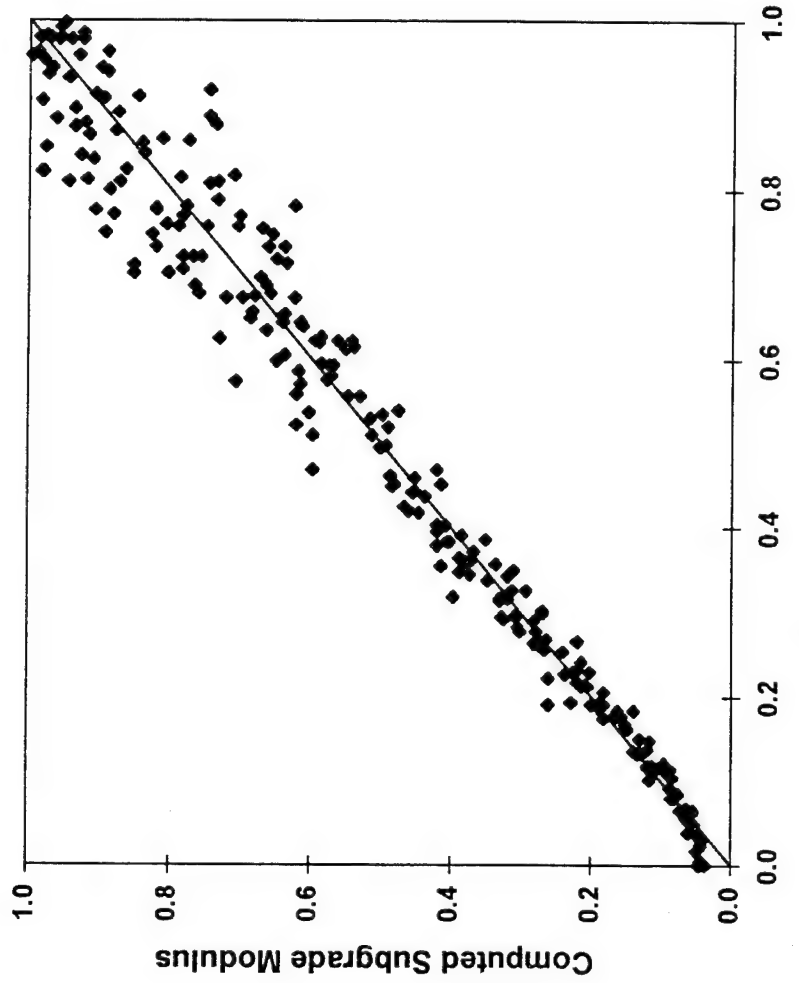


Figure 59. Normalized subgrade moduli backcalculated from "noisy" deflection basins

Determining the Network Architecture

Because the task of learning to map noisy data is more difficult for the network, the number of processing elements in the hidden layers had to be increased. Rather than use trial-and-error to determine the optimum number of neurons, a small parametric study was conducted. Referring back to Figure 52, the network's learning curves for this problem are fairly smooth and monotonic and the initial slope seems to be a fairly good predictor of the final error level. Assuming that this heuristic applies to networks trained with noisy data as well, a series of networks were created, each with more hidden-layer neurons than the last. These were then trained using just enough epochs to establish their initial learning rates and the network with the best initial learning rate was singled out for continued training.

Networks were created with 9, 12, 15, and 18 neurons in each of the two hidden layers. (Although the optimal network architecture probably has a different number of neurons in each of the hidden layers, the parametric study was made much simpler by using the same number of neurons in each layer.) Each network was initially trained for 200 epochs and the mean squared output errors plotted as a function of the number of hidden layer neurons (Figure 60). Each network was subsequently trained for an additional 100 epochs to confirm the initial findings. Those results are also plotted in Figure 60 along with the required processing time per training epoch. The improvement in accuracy

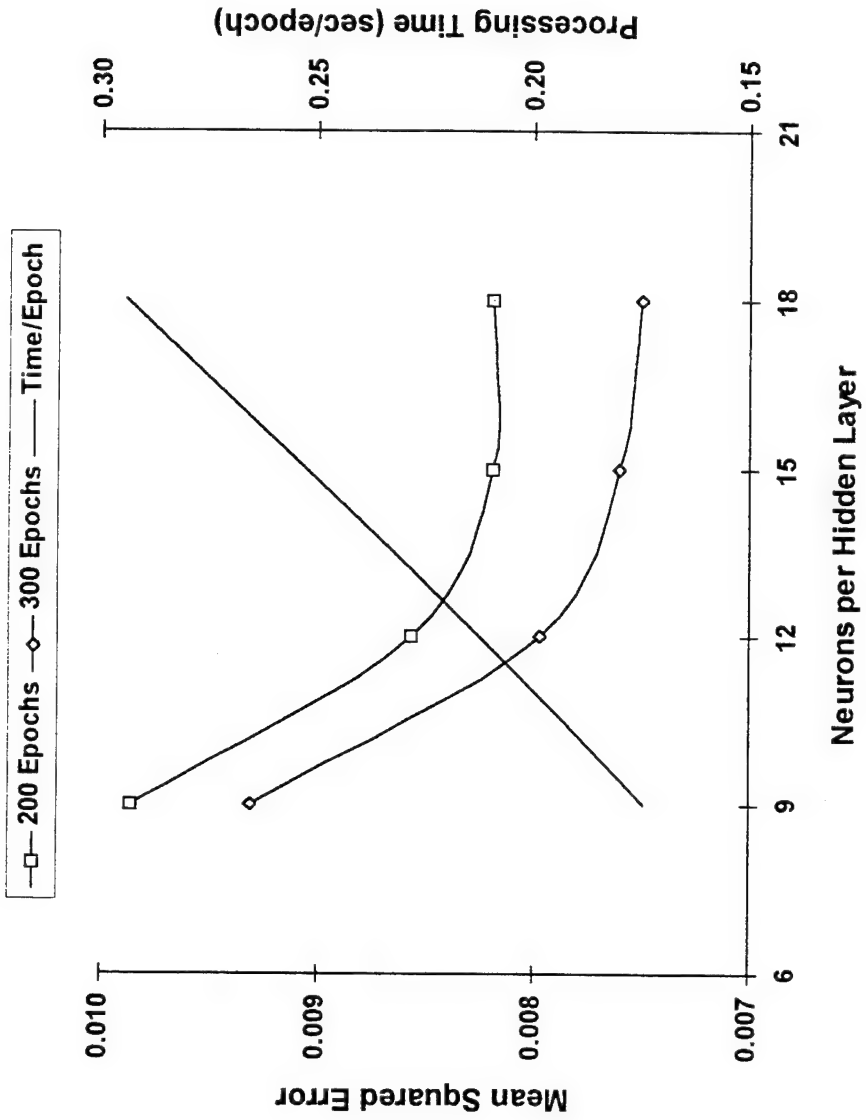


Figure 60. Network error and training time as a function of network size

afforded by the 15-neuron network over the 12-neuron network was sufficient to justify the 20 percent increase in training time; the very slight additional improvement afforded by the 18-neuron network, however, was not. Therefore, the network with 15 neurons in each hidden layer was chosen as having the optimum combination of training speed and training accuracy. This network will henceforth be referred to as the "Robust Network."

Training the Network

The Robust Network was trained for a total of 8000 epochs. Its training history is shown in Figure 61. Despite the fact that the curve for the testing set is not nearly as smooth as for the training set (one is based on an average of 250 values and the other is based on an average of 9,750 values), the two overlay each other. This shows that the network has learned the functional mapping rather than simply memorize the training set.

Notice that the final value of the mean squared error is about 0.005 for this network compared to 0.0007 for the network trained with noise-free data. This trade-off between accuracy and robustness is to be expected. Notice also that the network required twice as many epochs of training (8000 vs. 4000) to achieve a nearly constant mean squared error. This is also to be expected because the technique used to generate the random noise ensured that the network never saw the same basin twice. By comparison, the network trained using ideal deflection basins (henceforth referred to as the "Perfect Network")

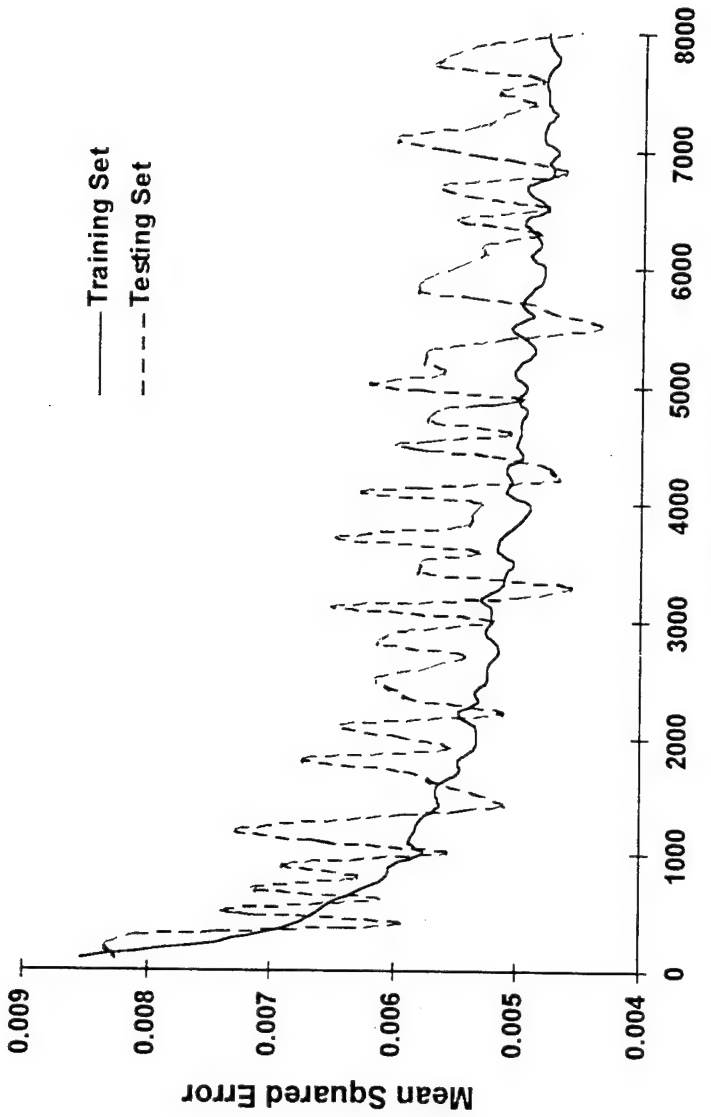


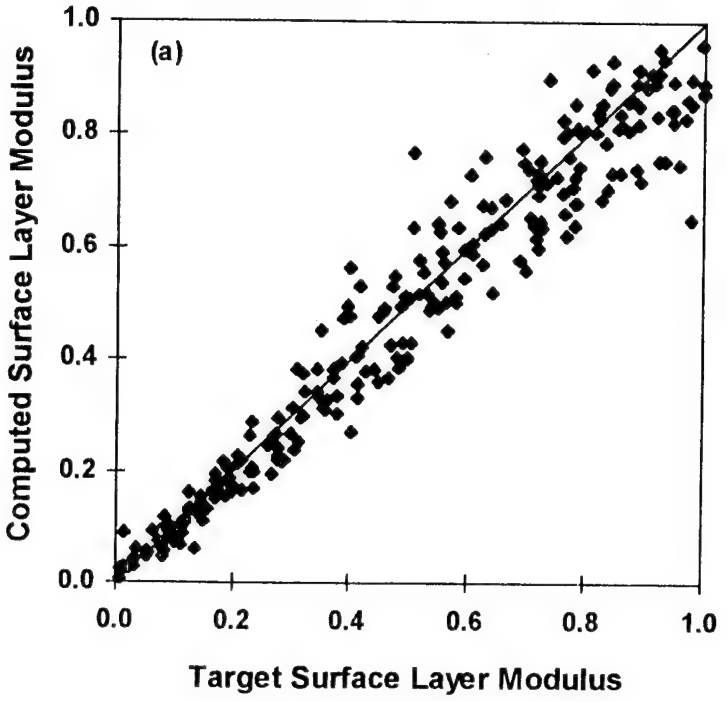
Figure 61. Training history for the robust network

saw each of those deflection basins 4000 times. With much less uncertainty in the training data, the Perfect Network converged much more quickly to the error surface minimum.

Comparison Between Calculated and Target Moduli

The backcalculation abilities of the Robust Network were determined using the same 250 "noisy" deflection basins as were used to generate the scatter plots in Figures 57-59. Figures 62a, 63a, and 64a show the results. A qualitative comparison between these figures and Figures 57-59 shows that the noise injection has produced a considerable improvement in the network's backcalculation ability.

Figures 62b, 63b, and 64b are scatter plots showing the results obtained using WESDEF on the same noisy deflection basins. The WESDEF moduli appear to exhibit as much scatter as, if not more than, the moduli computed with the neural network. This suggests that the scatter is mostly due to the inaccuracies in the deflections rather than a shortcoming in the neural network approach. A quantitative comparison of the predictive abilities of the Perfect Network, the Robust Network, and WESDEF is afforded by Figures 65–67, which show cumulative frequency distributions of the relative backcalculation errors for the surface, base, and subgrade moduli, respectively. These clearly illustrate the relative differences in accuracy of the three backcalculation programs.



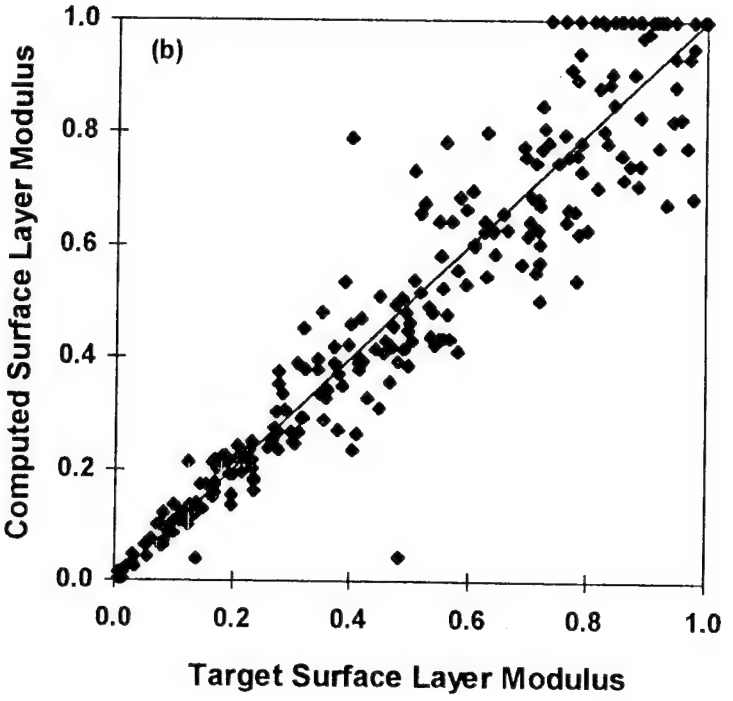


Figure 62. Normalized surface layer moduli from (a) robust network and (b) WESDEF

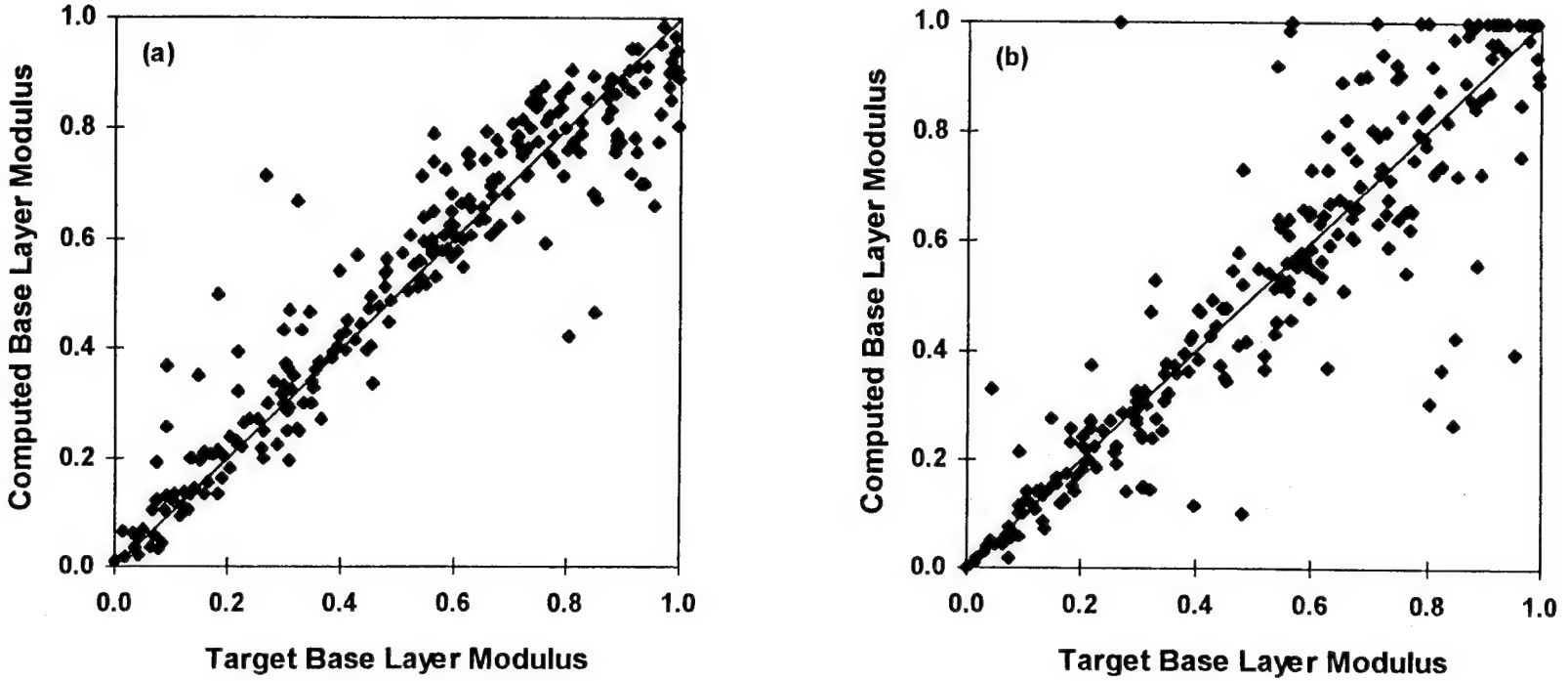


Figure 63. Normalized base layer moduli from (a) robust network and (b) WESDEF

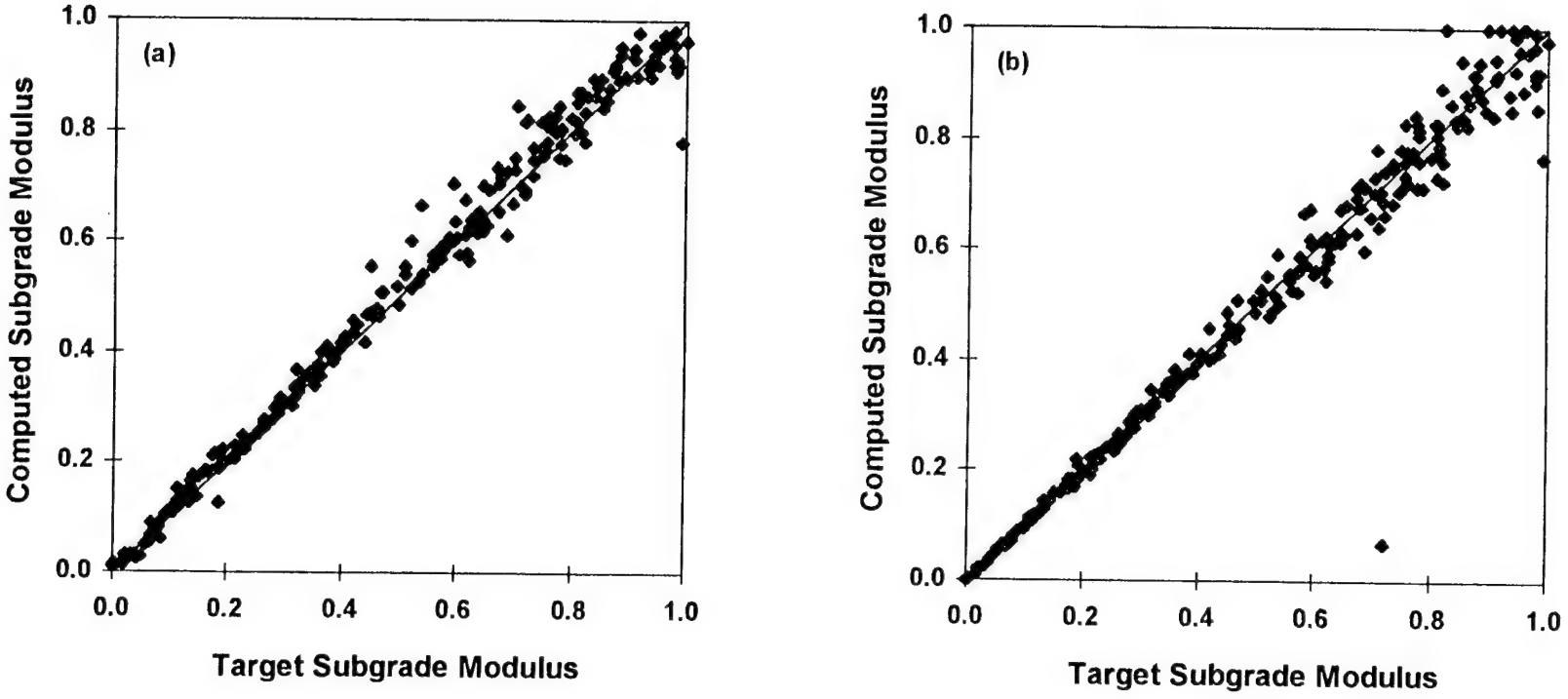


Figure 64. Normalized subgrade moduli from (a) robust network and (b) WESDEF

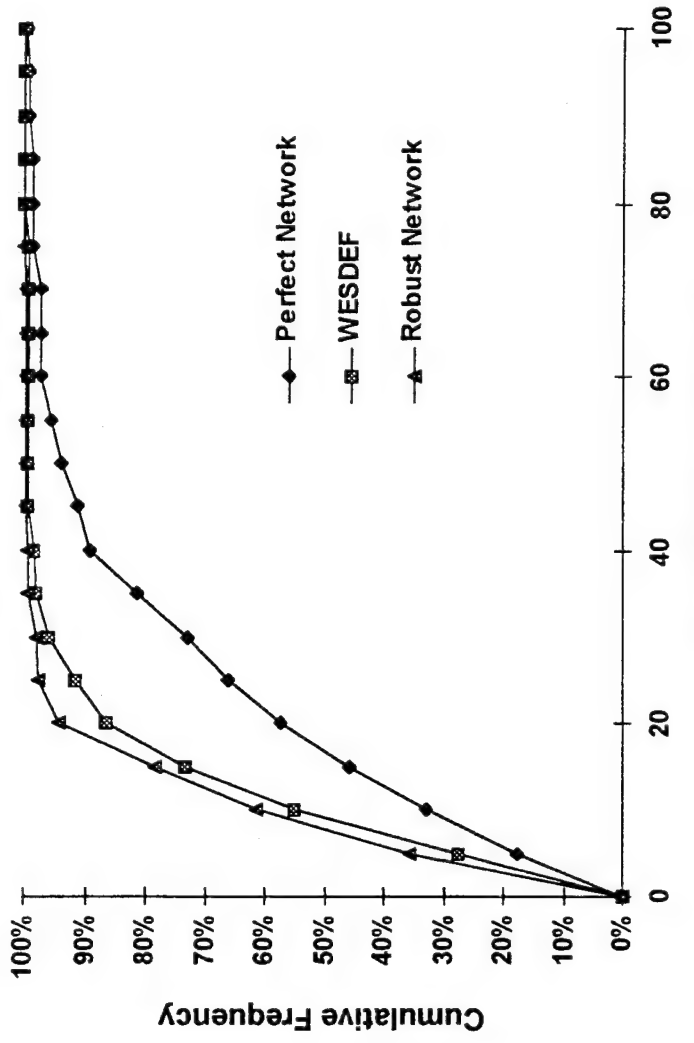


Figure 65. Cumulative frequency distributions of surface modulus errors

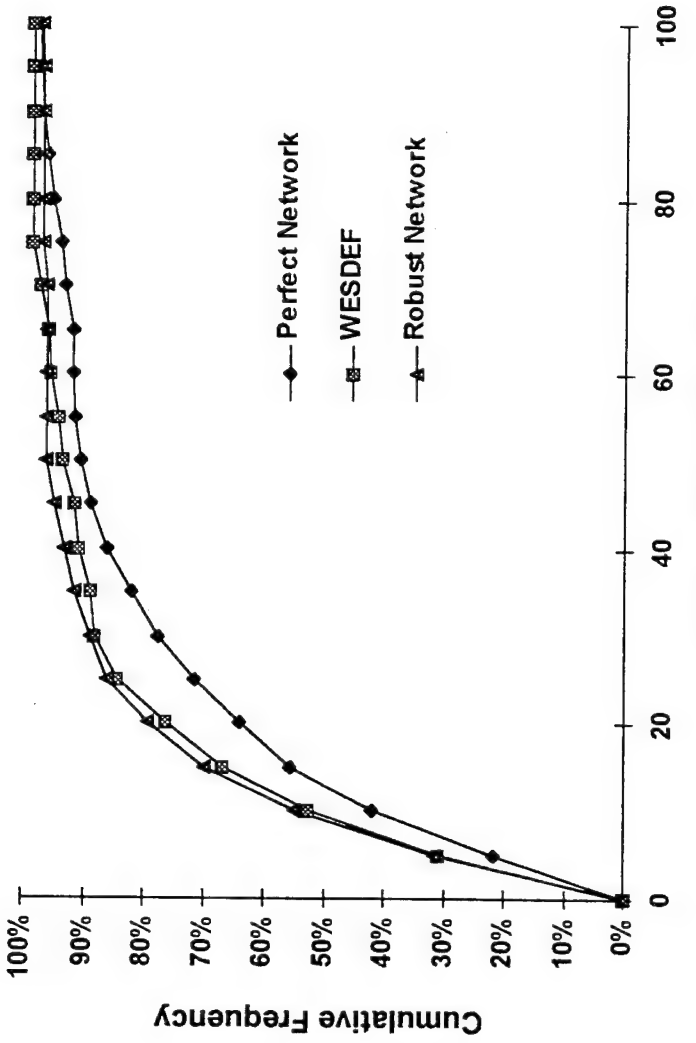


Figure 66. Cumulative frequency distributions of base modulus error

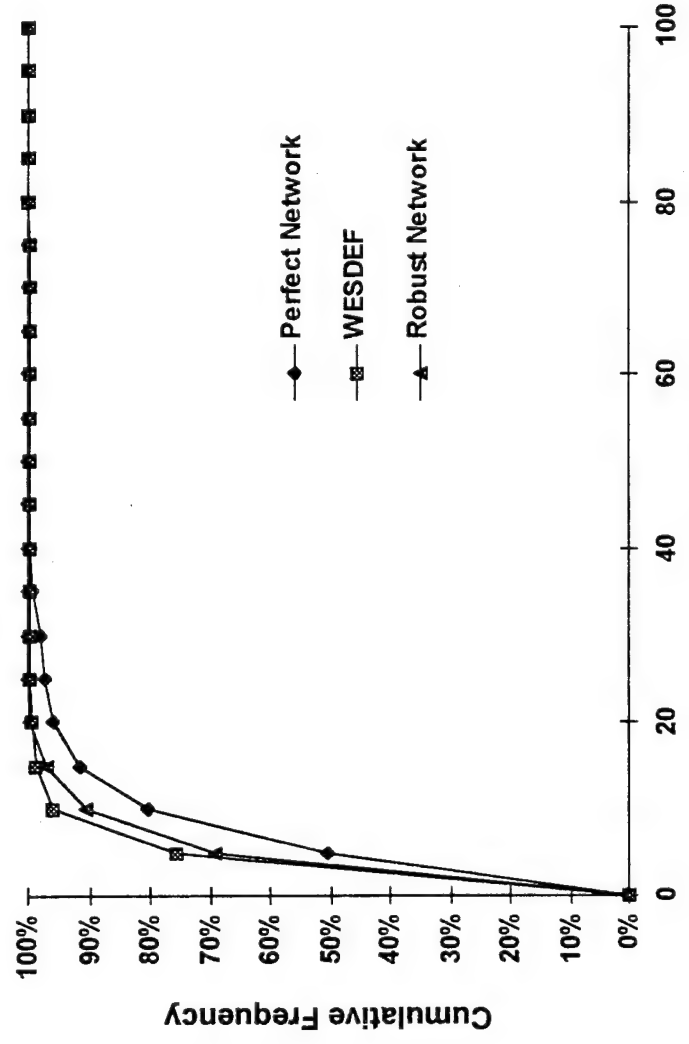


Figure 67. Cumulative frequency distributions of subgrade modulus error

Because much of the measurement error is the result of random noise, the accuracy of the backcalculated moduli can be improved considerably by replicating each test and averaging the results. For example, Irwin, Yang and Stubstad (1993) recommend that 3–5 replicates be obtained for each drop height. To illustrate this point, the Robust Network was retested with noisy deflection basins designed to represent an average of four replicated tests.

If the noise is truly random, its distribution should be Gaussian rather than uniform; therefore, a Gaussian distribution with a standard deviation of 0.04 mils was used to generate the random noise. The standard deviation of the Gaussian distribution was chosen so that approximately 95 percent of the noise (corresponding to ±2 standard deviations) would be less than the ASTM specification of 0.08 mils.

For each deflection basin in the independent test set, an amount of noise equal to the average of four random variates was added to each deflection. Figures 68–70 compare the target and computed moduli for the surface, base, and subgrade layers, respectively. These figures show considerably less scatter in the backcalculated moduli. Figure 71 shows the cumulative frequency distributions for the surface, base, and subgrade moduli. From this figure, it can be seen that the 80th-percentile errors for the three moduli are approximately 10, 15, and 5 percent, respectively. This is extremely good accuracy for moduli backcalculated from realistically noisy data.

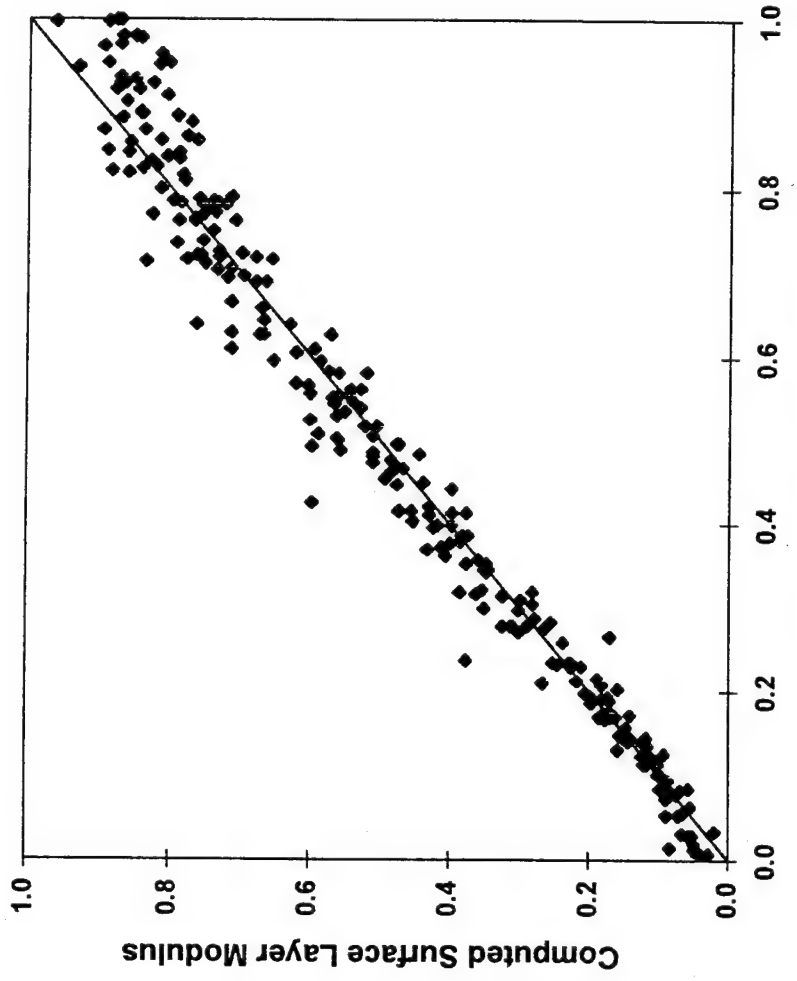


Figure 68. Normalized surface layer moduli backcalculated from replicated deflection basins

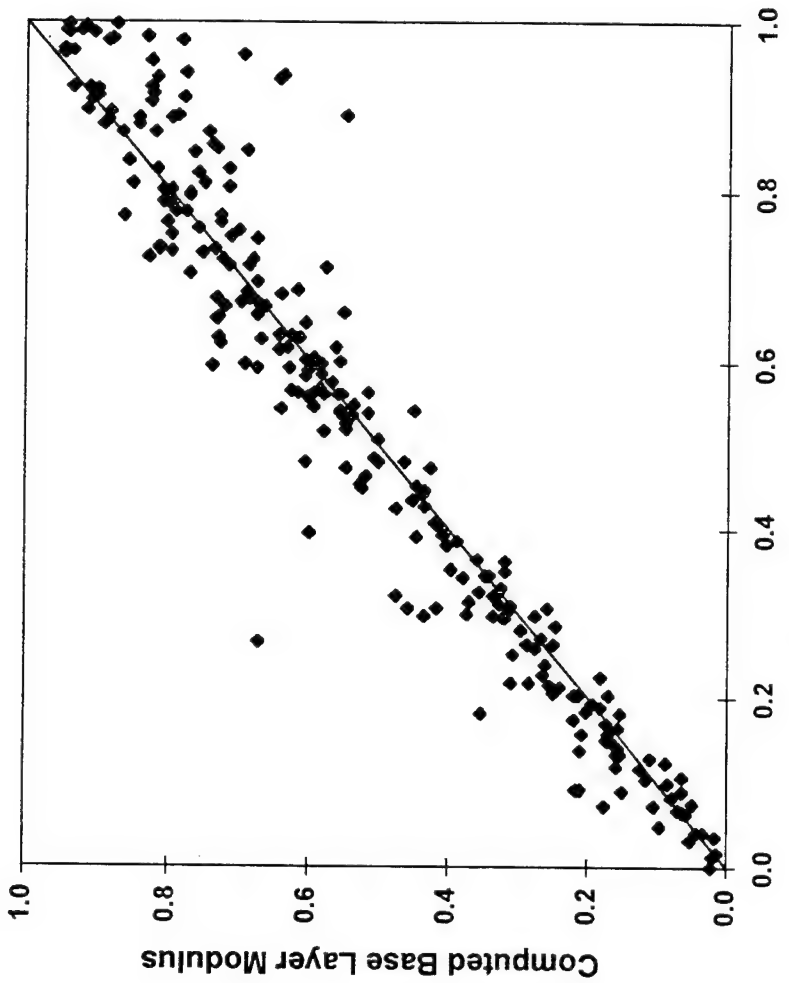


Figure 69. Normalized base layer moduli backcalculated from replicated deflection basins

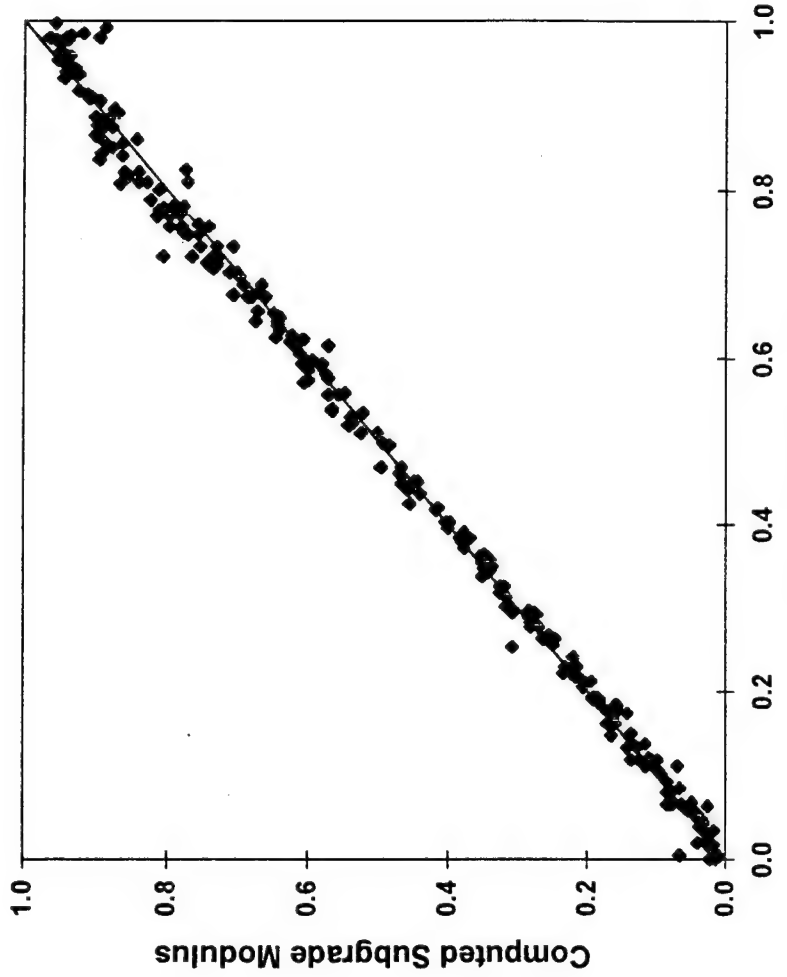


Figure 70. Normalized subgrade moduli backcalculated from replicated deflection basins

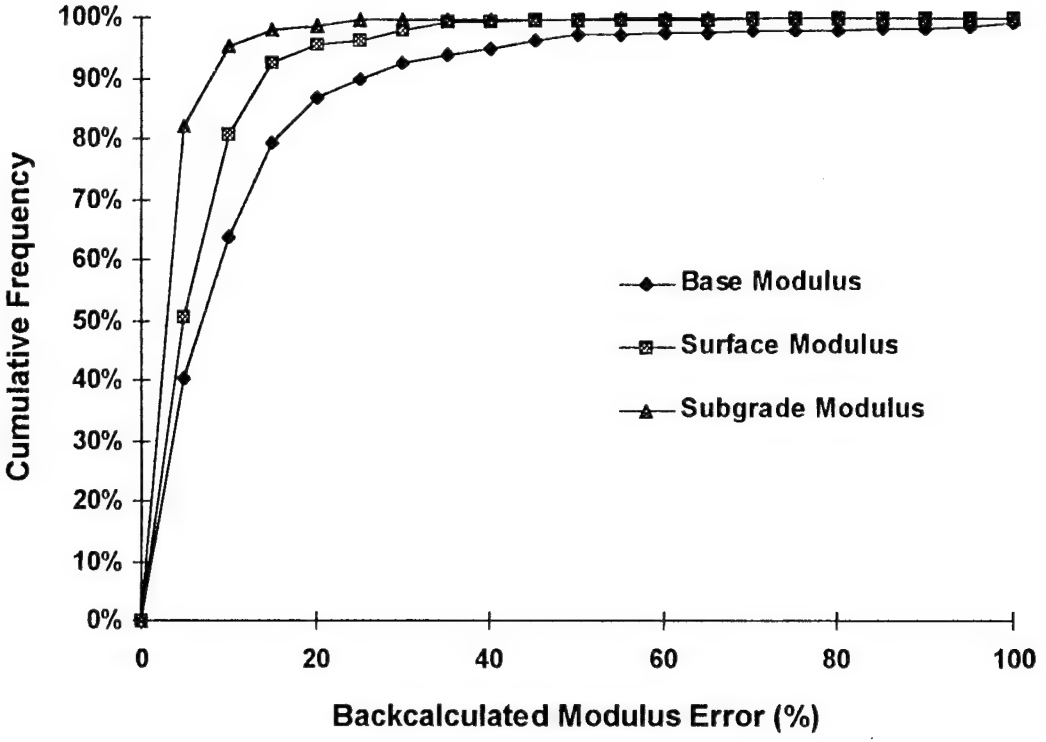


Figure 71. Cumulative frequency distributions of modulus errors from replicated deflection basins

Backcalculation from Experimental Deflection Basins

Having shown that neural networks can backcalculate pavement layer moduli from *synthetic* deflection basins with the same degree of accuracy as a traditional basin-matching program (WESDEF), the next logical step was to compare its accuracy on *experimental* data. This was accomplished using data from a blind test of several traditional basin-matching programs (Rada, Richter, and Stephenson, 1992). Sections A and B from that software evaluation exercise (shown in Figure 72) were selected because they are most similar to the three-layer, flexible pavements assumed in the network training. For Section A, the crushed limestone base and soil/aggregate subbase were combined to form a single base layer. In both cases, the subgrade was assumed to be semi-infinite because it was reported that no bedrock had been encountered at either test section within the top 20 ft.

The measured pavement deflections (Figure 73) were normalized to a load of 9000 lbs and propagated, along with the layer thicknesses, through the Robust Network. The same deflections and layer thicknesses were provided to MODULUS 4.0 (which uses the database approach) and WESDEF (which uses the gradient search approach). The backcalculated moduli from the Robust Network and the two basin-matching programs are shown in Table 7. For Section A, the neural network moduli were virtually identical to those provided by MODULUS 4.0 and WESDEF. For Section B, the neural network

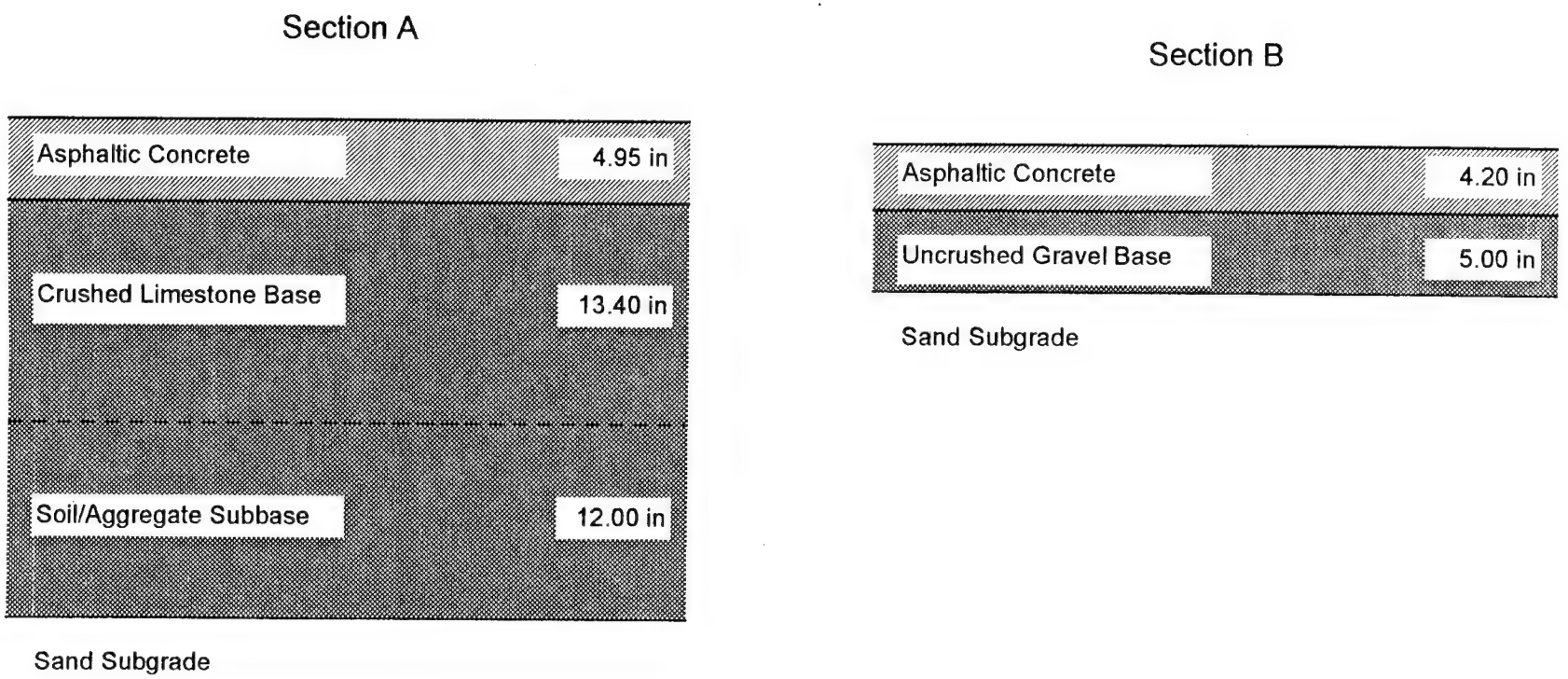


Figure 72. Pavement structures for SHRP test sections A and B

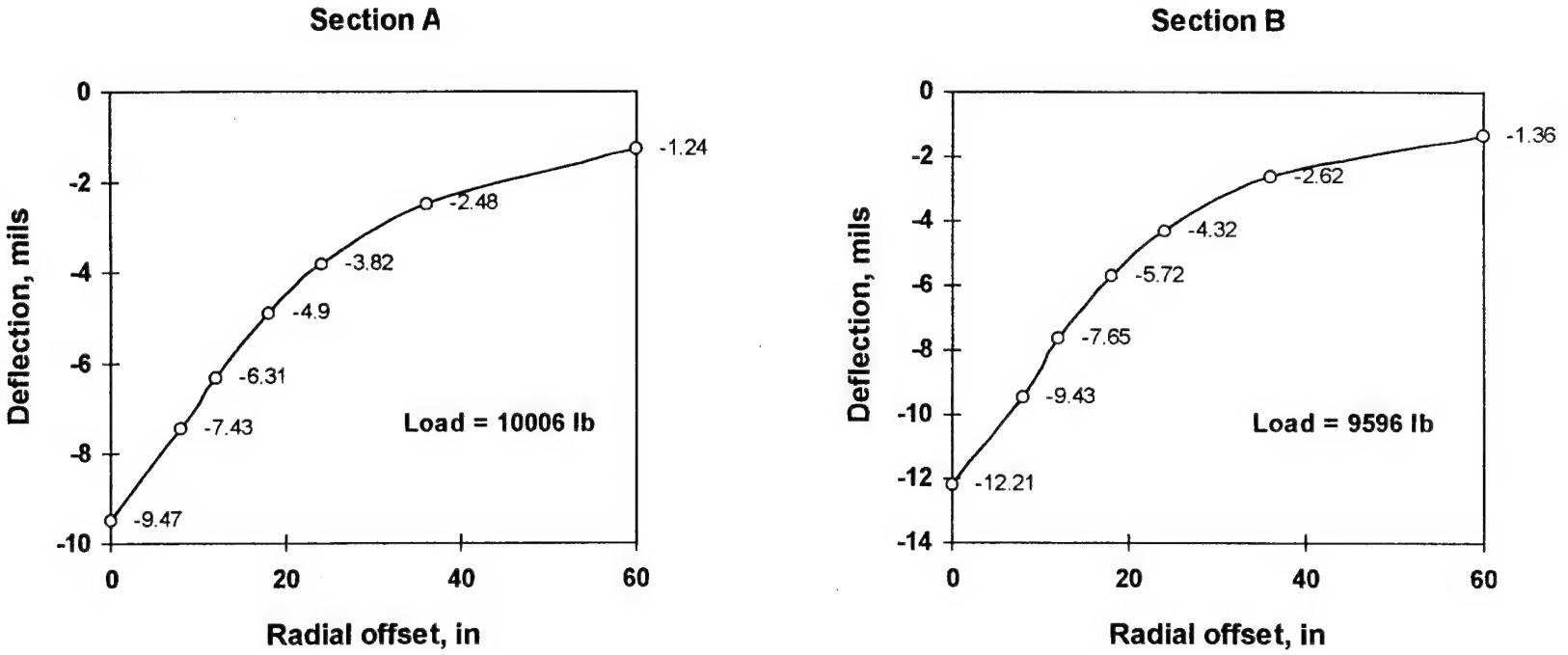


Figure 73. Experimental deflection basins from SHRP test sections A and B

Table 7. Moduli backcalculated from experimental deflection basins

SHRP Section	Pavement Layer	Backcalculated Moduli (ksi)		
		Neural Network	MODULUS 4.0	WESDEF
A	Asphalt	1294	1250	1317
	Base*	42	41	42
	Subgrade	32	30	31
B	Asphalt	855	921	918
	Base	53	56	46
	Subgrade	27	27	27

*Combination of crushed limestone base and soil/aggregate subbase

backcalculated a slightly lower asphalt modulus, but virtually identical base and subgrade moduli. Since the true moduli at the two test sections are not known, success can only be measured in comparison to the predictions produced by other programs; these results indicate that the neural network performed well on this experimental data.

Comparison of Resource Requirements

The primary advantage of using artificial neural networks is the speed with which pavement moduli can be backcalculated. To illustrate this point, a timing comparison was conducted between the Robust Network and WESDEF. In order to make the comparison as fair as possible, the WESDEF program was modified so it had the same input and output requirements as the neural network—the layer thicknesses and deflection basins were read into the program from one file and the backcalculated moduli were written out to another file. The modulus ranges required by WESDEF to bound the solution space were set to the maximum and minimum moduli from Table 2. (Referring back to Figures 62b, 63b, and 64b, this explains why so many of the plotted points are lined up along the top of the plot—WESDEF cannot backcalculate moduli outside of the established limits, so a normalized modulus of 1.0 is the largest possible value.)

Table 8 shows the processing times required by both WESDEF and the Robust Network to backcalculate the moduli for the 250 "perfect" deflection

Table 8. Comparison of processing times for 250 deflection basins

	Ideal Deflection Basins	Basins with Random Noise
Neural Network	0.9 sec	0.9 sec
WESDEF	25.0 min	37.5 min

basins and the 250 "noisy" deflection basins. The timing comparison was conducted on a 33-MHz 80486 personal computer. For the 250 basins with no added noise, WESDEF took 25 minutes to backcalculate all of the moduli. With the random noise added, WESDEF required 37.5 minutes. The neural network backcalculated each set of moduli in just 0.9 seconds.

The neural network processed the "noisy" data just as quickly as the noise-free data because the deflection inputs were propagated through the exact same network. (Had the Perfect Network been used on the noise-free deflection basins, instead, the processing time would have been slightly lower because the Perfect Network is slightly smaller than the Robust Network.) WESDEF, on the other hand, must iteratively seek a theoretical basin that matches the noisy experimental basin. This is a more difficult task than for noise-free basins, so the process took 50 percent longer.

Network Retraining Using Dynamic Deflection Basins

In the second phase of the research, the Robust Network was retrained using the deflection basins generated by the elastodynamic Green function solutions described in Chapter 5. As mentioned previously, there was no need to experiment with different network architectures in the second phase of the research because the mapping problem—from "noisy" deflection basins and pavement layer thicknesses to pavement layer moduli—was exactly the same as in the first phase of the study.

The exemplars in the dynamic training set were scaled in the same way as those in the static training set. The first 9,750 exemplars were used to train the network and the remaining 250 exemplars were reserved as the independent testing set. As in the first phase, random noise was added to the deflection inputs to make the network more robust. The noise was drawn, as before, from a uniform distribution with limits equal to the larger of 0.1 mils or two percent of the deflection.

Figure 74 illustrates the training progress of the retrained network (henceforth referred to as the Dynamic Network). Comparison with Figure 61 shows that the Dynamic Network trained to an error level almost identical to that of the Robust Network. As was done in Phase I, the Dynamic Network was tested using noisy deflection basins designed to simulate an average of four replicated tests. Figures 75–77 compare the target and computed moduli for the surface, base, and subgrade, respectively, and Figure 78 shows the corresponding cumulative frequency distributions of the network errors.

These results clearly show that the Robust Network was successfully retrained with the dynamic deflection basins to produce the Dynamic Network. Because the network architecture is the same, the Dynamic Network will be able to backcalculate moduli in exactly the same amount of time as the Robust Network. This means that the increased realism afforded by the dynamic model of the FWD was attained without paying a penalty in backcalculation time.

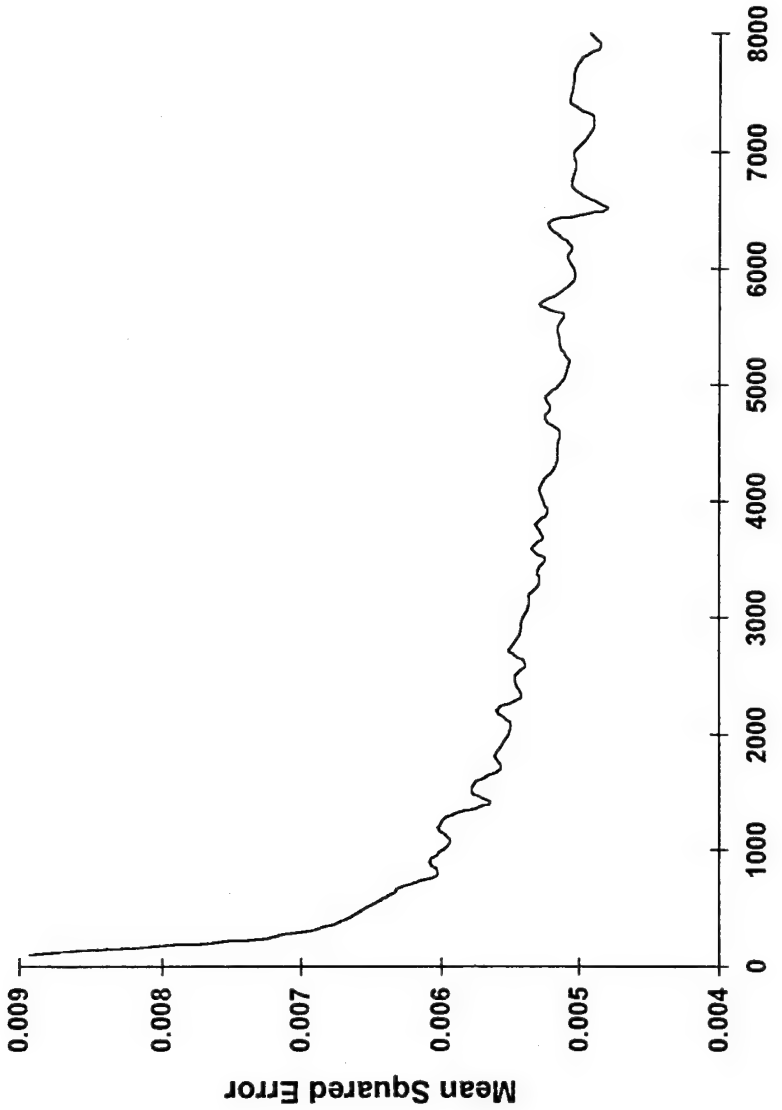


Figure 74. Training history of network trained with dynamic deflection basins

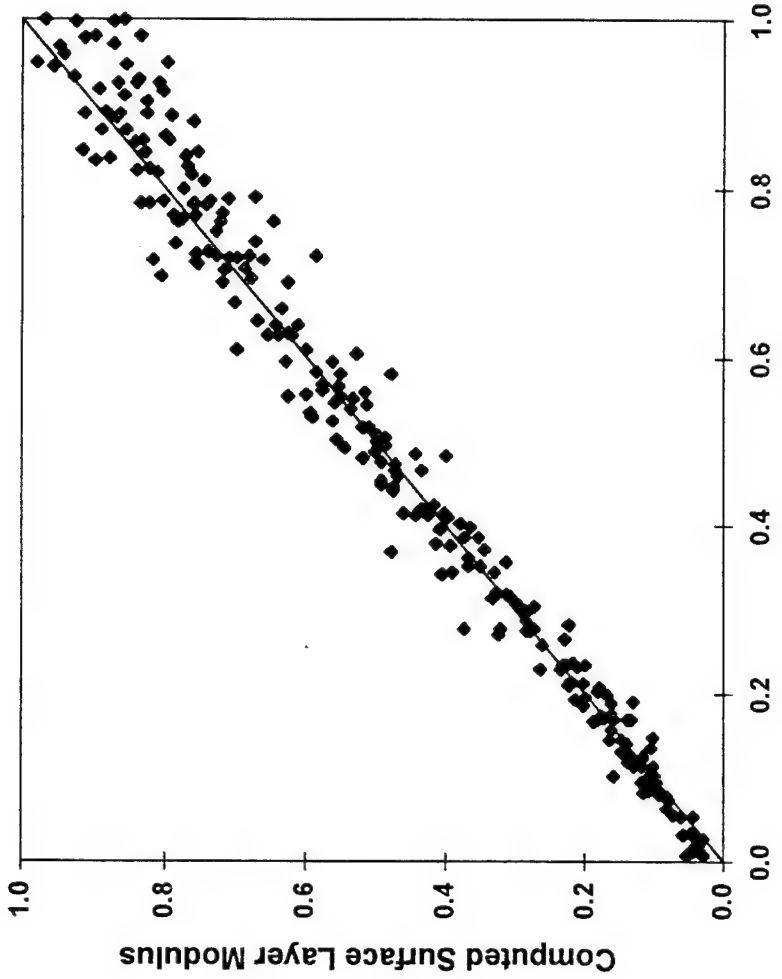


Figure 75. Normalized surface layer moduli backcalculated from dynamic deflection basins

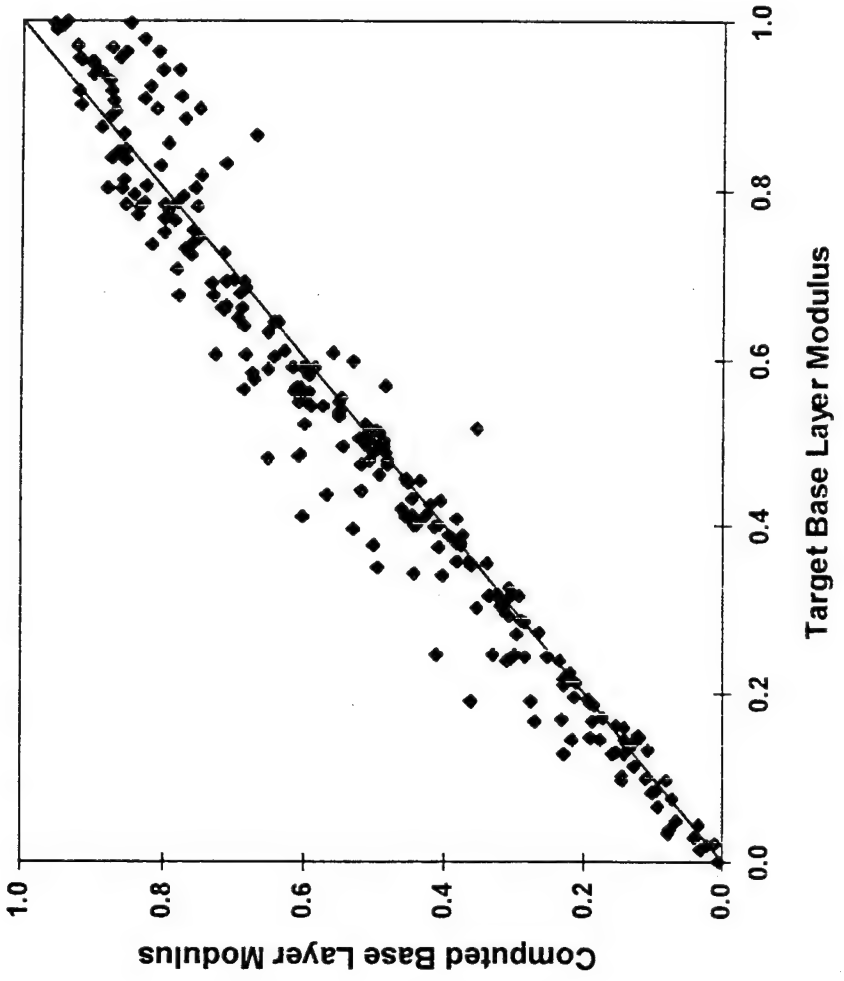


Figure 76. Normalized base layer moduli backcalculated from dynamic deflection basins

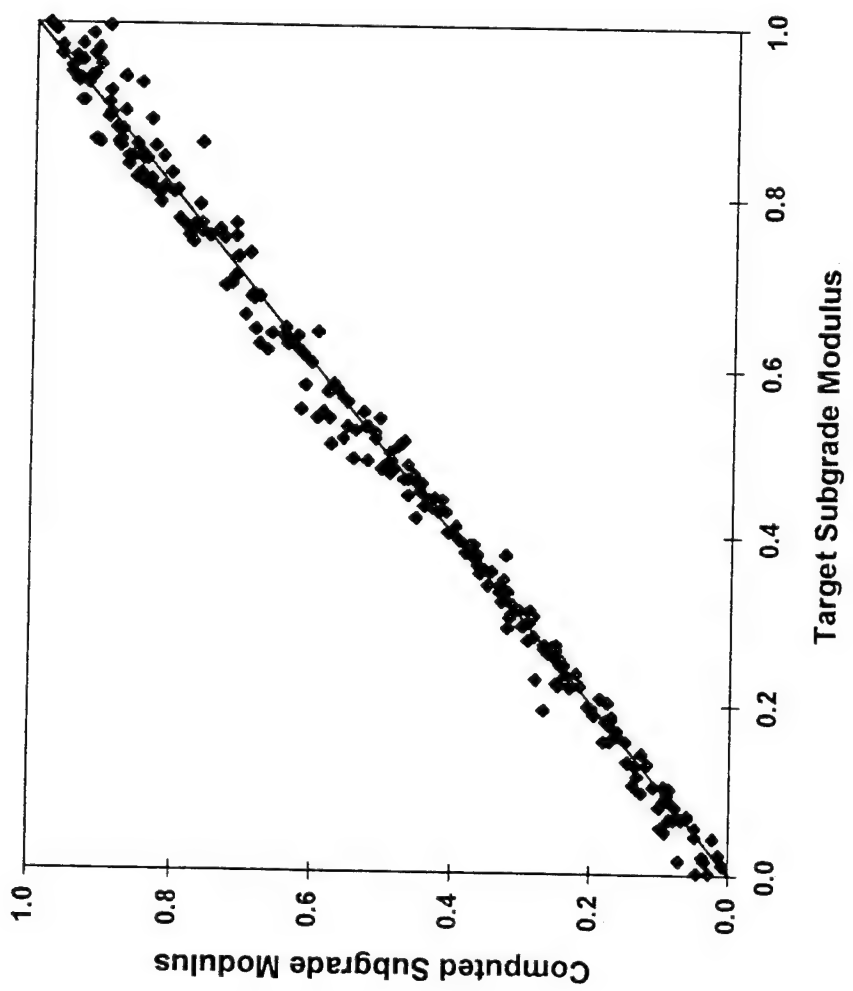


Figure 77. Normalized subgrade moduli backcalculated from dynamic deflection basins

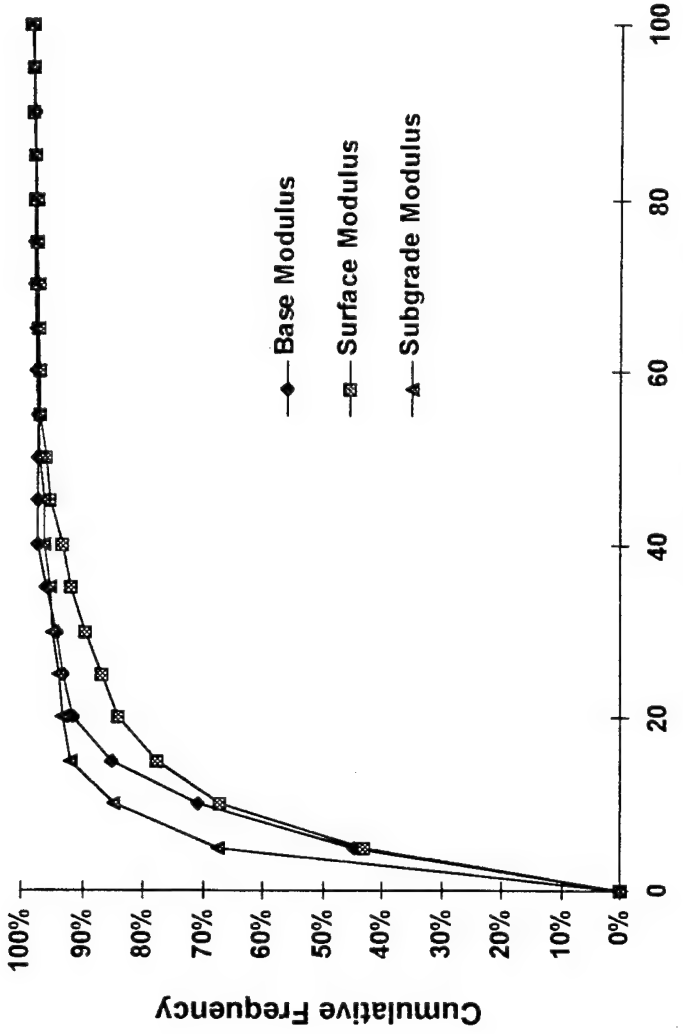


Figure 78. Cumulative frequency distributions of modulus errors from dynamic deflection basins

Summary

This chapter has described the training of neural networks capable of backcalculating pavement layer moduli from both ideal and noisy deflection basins generated using a static model of the FWD test. It has also described the retraining of one of those neural networks using deflection basins generated using an elastodynamic model of the FWD test.

The successful training of the neural network using ideal deflection basins (which has been called the Perfect Network) clearly shows that neural networks can be taught to solve complex, nonlinear inverse problems using training data generated by repeatedly solving the forward problem. This capability can easily be extended to a wide variety of inverse problems in the field of geophysics and elsewhere.

The neural network trained using noisy deflection data (which has been called the Robust Network) was shown to be significantly more robust against the measurement errors common in experimental data. That neural network has been shown to backcalculate moduli with an accuracy equal to or better than the conventional basin-matching program WESDEF, but at a speed three orders of magnitude faster.

The successful retraining of the Robust Network using synthetic deflection basins generated using the elastodynamic Green functions described in Chapter 5 has shown that the computational efficiency of the

neural network can be preserved while using more realistic, albeit more complex, solutions. This retrained network (which has been called the Dynamic Network) has two primary advantages over the original version: 1) because it is based on a more realistic model of the pavement's response to the FWD loads, it should provide a more realistic estimate of the *in situ* moduli, and 2) because the dynamic pavement response model is relatively insensitive to bedrock depth, it should be able to better backcalculate moduli when bedrock depth is either unknown or had been determined inaccurately. Most importantly, because the same network architecture was used for both the Robust Network and the Dynamic Network, these advantages can be realized without an increase in processing time.

CHAPTER 7

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Background

The structural integrity of roads and airfields is determined in large part by the load-deflection properties of the material layers that make up the pavement system. Those properties can be measured *in situ* using nondestructive testing (NDT) techniques. The falling-weight deflectometer (FWD) test is one of the most widely used NDT tests. The FWD measures the dynamic response of a pavement system to a transient load applied at the pavement surface. The evaluation of its results generally entails backcalculating the *in situ* pavement layer moduli from the peak measured deflections at several radial offsets from the load.

A major limitation of existing techniques for backcalculating pavement layer moduli from FWD results is that they are computationally inefficient—they all involve multiple calculations of pavement response for a variety of presumed moduli. This not only makes them tedious to use, but also constrains them to employ simplified static analyses of pavement response that can be

computed relatively quickly. Studies have shown that significant errors in the backcalculated pavement moduli can accrue from the static analysis of the inherently dynamic FWD test.

Artificial neural networks represent a fundamentally new approach to the backcalculation of pavement layer moduli from FWD test results. An artificial neural network is a highly-interconnected collection of relatively simple processing elements that can be trained to approximate a complex, nonlinear function through repeated exposure to examples of the function. In the context of backcalculation, a neural network can be trained to approximate the inverse (backcalculation) function by showing it synthetic test results generated by repeated application of the forward problem solution. The forward problem solution can employ either a static or a dynamic analysis of the pavement response. Both were used in this study.

Summary

The goal of the first phase of this research was to show that it is possible to perform real-time backcalculation of pavement layer moduli using artificial neural networks. In order to allow a direct comparison between the neural network approach and a conventional basin-matching approach, an artificial neural network was trained using synthetic deflection basins generated by the same static, layered elastic computer program as is used in the conventional backcalculation program WESDEF. That network, called the Perfect Network,

was shown to accurately backcalculate layer moduli when presented with perfect (i.e., noise-free) deflection basins. When the Perfect Network was presented with noisy deflection basins more typical of those measured in the field, however, its accuracy was seen to be poor. To remedy this, another neural network was trained using deflection basins contaminated with random noise. This neural network, called the Robust Network, was shown to have very good accuracy when tested with synthetic deflection basins designed to simulate replicated field tests. When compared directly to WESDEF, the Robust Network was shown to be as accurate, if not more accurate, and 2500 times faster. This speed gain was more than sufficient to achieve real-time backcalculation.

The goal of the second phase of the research was to improve on the accuracy of the Robust Network by retraining it with synthetic deflection basins based on a dynamic analysis of the pavement response. The dynamic analysis provides a much better approximation of the actual test conditions. It also avoids certain problems, such as an excessive sensitivity to the assumed bedrock depth, that are inherent in the static analysis. Because the computational efficiency of a trained neural network is completely independent of the computational complexity of the program used to generate its training set, it was possible to retrain the network without increasing its processing time. Though it took 20 times longer to create the training set, the retrained

neural network could backcalculate moduli as quickly as the one trained using a static analysis. Contrast this with a conventional gradient search program: the conventional program must repeatedly solve the more-complex dynamic problem to obtain a solution. Based on the relative amounts of time needed to create the static and dynamic training sets, WESDEF would run 20 times slower if its static solution were to be replaced by the elastodynamic Green function solution used here. This means that the neural network trained using dynamic deflection basins would likely be 50,000 times faster than its comparable basin-matching program. There is no other inversion technique that can match this performance.

Conclusions

In the first phase of the research, it was clearly shown that neural networks can be taught to solve complex, nonlinear inverse problems by solving the forward problem for a wide variety of inputs and using the neural network to map the results back onto the original inputs. Together, the neural network and the forward problem solution form a closed loop (Figure 19). Half of the loop implements the forward problem while the other half (the neural network half) implements the inverse problem. This is an especially important conclusion because the same technique can be applied to many geophysical inversion problems—not just the backcalculation of pavement moduli.

The first phase of the research also showed that neural networks could, indeed, backcalculate pavement layer moduli from FWD data in real time. Previously, the FWD could generate data faster than it could be analyzed. This limited the usefulness of the FWD for routine pavement evaluation. With the backcalculation bottleneck eliminated, the test data can be analyzed much faster and FWD tests can be performed more frequently without overwhelming the data analysts and engineers.

In the second phase of the research, it was shown that neural networks could be trained using synthetic test data generated using more realistic, albeit more complex, dynamic models of the FWD test without losing their ability to backcalculate in real time. This can increase the accuracy of the backcalculated moduli without the usual speed penalty paid with conventional backcalculation techniques.

Finally, real-time backcalculation opens up the possibility of performing both the data acquisition and the analysis in real time. It would no longer be necessary to close a traffic lane and divert traffic to perform a test. This would alleviate both the direct costs of traffic control and the indirect costs of commuting delays. Furthermore, the work crews performing the tests would be at much less risk if they did not have to work outside their vehicle in the middle of a traffic lane. Eventually, it may be possible to perform the entire test at a reasonable travel speed from the driver's seat of the test vehicle.

Recommendations

This research was confined to three-layer flexible pavement systems with an infinite subgrade thickness. By using a dynamic analysis of pavement response, subgrade thickness is eliminated from consideration for depths to bedrock in excess of roughly ten feet. For shallower bedrock, however, the resonance of the pavement system will be sufficient to affect the peak deflections. For those pavement systems, the subgrade thickness must somehow be considered in the backcalculation. If the subgrade thickness is known *a priori*, or can be estimated using empirical techniques such as that developed by Rohde (1990), it can easily be accommodated as another thickness input to the neural network. Thus, for shallow bedrock, another network should be trained using subgrade thickness as input.

Another area where enhancements can be made is in the modeling of pavement systems with more than three layers. Additional networks for use with four- and five-layer pavement systems would be desirable. It may not be feasible, however, to backcalculate even four moduli with any accuracy. Though it is theoretically possible to backcalculate seven variables from the seven deflection measurements, some redundancy is required to compensate for the noise inherent in the data measurements. For this reason, it would be difficult to expand the analysis to more than three moduli or to replace the

elastic moduli with time- or stress-dependent material properties. Seven peak deflections is simply not enough data.

Yet another area with considerable room for improvement is in the constitutive modeling of the pavement layers. It is perhaps too great a simplification to treat the pavement layers as linear-elastic continua. The response of most pavement and subgrade materials is, in fact, both rate- and stress-dependent. Furthermore, the materials have considerably different tensile and compressive properties (often characterized by a complete absence of tensile strength). Some of these problems can be overcome by using more sophisticated models of pavement response. Unfortunately, the same problem arises here as with the modeling of additional pavement layers—with just seven peak deflections, there is simply not enough information to backcalculate the additional parameters needed to describe rate- or stress-dependency.

Some of the aforementioned limitations can be overcome by using more of the available data. The Dynatest FWD actually records the surface deflection *histories*; however, only the peak deflections are commonly used for analysis. As shown by Foinquinos, Roesset, and Stokoe (1993a), the depth to bedrock can be determined from the resonant frequency of the pavement system if the complete deflection histories are available. It may also be possible to establish viscoelastic material properties from the time-dependent experimental data. Unfortunately, backcalculating pavement layer properties

from complete deflection histories would be very time-consuming using traditional iterative approaches because of the volume of data and the number of independent variables. Because neural networks can process large amounts of data very quickly, they provide a realistic means of accomplishing that task. A neural network could, for example, be taught to backcalculate pavement layer properties from the peak deflections and the time lags between peaks, or the peak deflections and the rise times of each deflections pulse. For that matter, a neural network could be trained using discretized deflection pulses or a discrete FFT of the deflection pulses. Regardless of the data chosen for use in the analysis, the same training concepts that were developed here would be used to train the networks.

The Potential for Data Fusion

Even more promising than the use of complete deflection histories is the use of more than one type of NDT device. There are many different types of NDT devices currently used in pavement testing. Each is based on a different geophysical principle and each has advantages and disadvantages relative to the others. Much better estimates of pavement layer properties—or estimates of additional pavement layer properties—could be obtained by merging, or "fusing", data from two or more platforms. By using complementary geophysical techniques, the strengths of each platform could be used to best advantage while their weaknesses could be compensated for by the other platforms.

For example, the FWD could be combined with ground penetrating radar to allow backcalculation of both the pavement layer moduli and the pavement layer thicknesses, thus eliminating the need for measured or estimated layer thicknesses. Another example would be to combine an FWD Analysis with a Spectral Analysis of Surface Waves. This would provide information about both the small-strain and large-strain elastic moduli of the pavement layers and help to determine the stress-dependency of the material properties.

Unfortunately, multiple NDT platforms would produce a large volume of experimental data. That data would overwhelm conventional backcalculation programs. Neural networks, because they can process large amounts of data quickly, have the potential to make backcalculation from fused data possible.

Because artificial neural networks are still a relatively new data analysis tool, their capabilities are still being explored and expanded. This research has focused on one type of neural network—the multi-layer, feed-forward network—because of its capabilities for functional approximation. There are other types of artificial neural network that are better suited to tasks such as pattern recognition and minimization. These need to be explored for their abilities to preprocess some of the accumulated data prior to the backcalculation. This is more than just the conclusion of a research project; it is the beginning of an entirely new way of performing data reduction and analysis.

APPENDIX A

ARTIFICIAL NEURAL NETWORK TRAINING PROGRAM

```
      PROGRAM BACKPROP
C     ------------------------------------------------------------------
C                                   BACKPROP
C
C                                     by
C
C                                 Roger W. Meier
C                          USAE Waterways Experiment Station
C                                3909 Halls Ferry Road
C                              Vicksburg, MS  39180-6199
C
C     This program trains a standard backpropagation-style artificial
C     neural network using the Generalized Delta Rule with a momemtum
C     term added.
C
C     The program makes NEpoch passes through the training set, which
C     consists of (NTrain+NTest) input/output pattern pairs read into
C     memory from the ASCII data file 'training.data'. The 1st NTrain
C     input/output pairs are used to train the network. The remaining
C     NTest pairs are used to monitor training progress. Every NPrint
C     passes through the dataset, target and computed results for the
C     NTest testing pairs are output to ASCII file 'training.results'
C     and the mean-squared output errors for the training and testing
C     sets are output to ASCII file 'training.progress'. Additionally
C     the current values of the interconnection weights are output to
C     ASCII file 'training.weights'. The training progress file grows
C     with each successive output while the results and weights files
C     are overwritten each time they are accessed.
C
C     This program allows networks to be trained incrementally. If an
C     existing training weights save file is located, it will be used
C     as the starting point for additional network training.
C
C     The maximum number of input/output pairs that can be read in is
C     specified by parameter NPairs, the maximum number of processing
C     elements in any one network layer is specified by the parameter
C     NMax, and the maximum number of hidden layers is set at two.
C
C     ------------------------------------------------------------------
C                           Network Training Inputs
C
C     NTrain ... number of input/output pairs used for training
C     NTest ... number of input/output pairs used for testing
C     NEpoch ... number of passes through the training/testing data
C     NPrint ... frequency of training progress printouts (to file)
C     Alpha ... initial learning rate for the generalized delta rule
C     DAlpha ... multiplier used to decrease learning rate each epoch
C     Beta ... initial momentum factor for the generalized delta rule
C     DBeta ... multiplier used to decrease momentum factor each epoch
C
C                          Network Architecture Inputs
C
C     NLayer ... number of layers (hidden and output) in the network
C     NIN ... number of network inputs (neurons in the input layer)
C     Nodes() ... number of neurons in the hidden and output layers
C     FMin ... minimum output of the sigmoidal logistic function
```

```
C       FMax ... maximum output of the sigmoidal logistic function
C       FGain ... gain of the sigmoidal logistic function
C
C                    Network Training/Testing Inputs/Outputs
C
C       XIN(,) ... array of training/testing input patterns
C       XOUT(,) ... array of training/testing output patterns
C       YOUT() ... array containing a calculated output pattern
C       ERROR() ... array of (calculated-target) output errors
C       MSE() ... array with training/testing mean-squared errors
C
C      ---------------------------------------------------------------

       Parameter (NMax=15,NPairs=1000)

       Real MSE(2)
       Logical Resume,Output
       Integer Epoch,Loop,Pick,Last,Count
       Dimension XIN(NMax,NPairs),XOUT(NMax,NPairs)
       Dimension YOUT(NMax),Error(NMax),SqrErr(2)
       Common /Netwrk/ NLayer,Nodes(0:3)
       Common /Weight/ W(NMax,NMax,3),B(NMax,3),Y(NMax,0:3)
       Common /Trains/ Alpha,DAlpha,Beta,DBeta,DW(NMax,NMax,3),DB(NMax,3)

C ... Read the setup and network information from 'stdin'

       Call INPUT (NTrain,NTest,NEpoch,NPrint,NIN,NOUT)

C ... Echo the setup and network information to 'stdout'

       Call ECHO (NTrain,NTest,NEpoch,NPrint,NIN,NOUT)

C ... Read the entire training data input file

       Call GETDATA (XIN,NIN,XOUT,NOUT,Count)

       If (Count.LT.NTrain) STOP'ERROR: Not enough data pairs to train'

       NTest = MIN(NTest,Count-NTrain)

C ... Check for an existing training weights save file

       Inquire (File='training.weights',Exist=Resume)

C ... If a training weight file already exists, read in the weight array
C ... and resume training; otherwise, initialize the weight array.

       If (Resume) Then
         Open (2,File='training.weights',Form='FORMATTED',Status='OLD')
         Read (2,*) Last,Alpha,Beta,B,W
       Else
         Open (2,File='training.weights',Form='FORMATTED',Status='NEW')
         Last = 0
         Call RESET
       End If
```

```
C ... Open the training output files and position their file pointers

      Open (3,File='training.progress',Form='FORMATTED')
      If (Resume) Call ADVANCE (3)
      Open (4,File='training.results',Form='FORMATTED')

C ... Loop through the training set NEpoch times

      Do 5 Epoch = Last+1,Last+NEpoch

        Output = (Epoch.EQ.Last+NEpoch) .OR. (MOD(Epoch,NPrint).EQ.0)
        If (Output) Rewind (4)

C ..... Initialize the error statistics to zero

        SqrErr(1) = 0.0
        SqrErr(2) = 0.0

C ..... Loop through the NTrain training pairs

        DO 2 Loop = 1,NTrain

C ....... Select an input pattern at random

          If (.True.) Then
             Pick = MIN(1+INT(NTrain*RANDOM(3)),NTrain)
          Else
             Pick = Loop
          End If

C ....... Propagate the input pattern through the network

          CALL ANN (XIN(1,Pick),NIN,YOUT,NOUT)

C ....... Compute the resulting output errors

          Do 1 K = 1,NOUT
            Error(K) = XOUT(K,Pick) - YOUT(K)
    1     Continue

C ....... Accumulate the output error statistics

          CALL STATS (Error,NOUT,SqrErr(1))

C ....... Adjust the weights using backpropagation algorithm

          CALL TRAIN (Error,NOUT)

    2   Continue

C ..... Save the network's current weights and biases

        If (Output) Then
           Rewind (2)
           Write (2,*) Epoch,Alpha,Beta,B,W
        End If
```

```
C ..... Loop through the NTest testing pairs

        DO 4 Loop = NTrain+1,NTrain+NTest

C ....... Propagate an input pattern through the network

          CALL ANN (XIN(1,Loop),NIN,YOUT,NOUT)

C ....... Compute the resulting output errors

          Do 3 K = 1,NOUT
            Error(K) = XOUT(K,Loop) - YOUT(K)
    3     Continue

C ....... Accumulate the output error statistics

          CALL STATS (Error,NOUT,SqrErr(2))

C ....... Write the expected and realized outputs to file

          If (Output) Write (4,*) (XOUT(K,Loop),YOUT(K),K=1,NOUT)

    4   Continue

C ..... Compute the average MSE for training and testing

        MSE(1) = SqrErr(1)/NOUT/MAX0(NTrain,1)
        MSE(2) = SqrErr(2)/NOUT/MAX0(NTest,1)

C ..... Report on the training progress

        If (Output) Then
          Write (*,*) Epoch,Alpha,Beta,MSE(1),MSE(2)
          Write (3,*) Epoch,Alpha,Beta,MSE(1),MSE(2)
        End If

C ..... Adjust the learning rate and momentum for the next pass

        Alpha = Alpha*DAlpha
        Beta = Beta*DBeta

    5 Continue

C ... Close all input and output files

      Close (2)
      Close (3)
      Close (4)

      STOP
      END

C     ---------------------------------------------------------------

      Subroutine INPUT (NTrain,NTest,NEpoch,NPrint,NIN,NOUT)
```

```
C     This FORTRAN routine reads in the network geometry descriptors and
C     the coefficients for the generalized delta rule from an ASCII file.

      Parameter (NMax=15)

      Common /Netwrk/ NLayer,Nodes(0:3)
      Common /XferFn/ FMin,FMax,FGain,FRange
      Common /Trains/ Alpha,DAlpha,Beta,DBeta,DW(NMax,NMax,3),DB(NMax,3)

C ... Read in training/testing information

      Read (5,*) NTrain
      Read (5,*) NTest
      Read (5,*) NEpoch
      Read (5,*) NPrint
      Read (5,*) Alpha
      Read (5,*) DAlpha
      Read (5,*) Beta
      Read (5,*) DBeta

C ... Read in network dimensions

      Read (5,*) NLayer
      If (NLayer.GT.3) STOP' Too many layers in this network'

      Read (5,*) NIN
      IF (NIN.GT.NMax) STOP' Too many neurons in the input layer'

      Do 1 I = 1,NLayer
      Read (5,*) Nodes(I)
    1 IF (Nodes(I).GT.NMax) STOP' Too many neurons in this layer'

      Nodes(0) = NIN
      NOUT = Nodes(NLayer)

C ... Read in transfer function parameters

      Read (5,*) FMin
      Read (5,*) FMax
      Read (5,*) FGain

      FRange = FMax - FMin

      Return
      End

C     ---------------------------------------------------------------

      Subroutine ECHO (NTrain,NTest,NEpoch,NPrint,NIN,NOUT)

C     This FORTRAN routine outputs the network geometry descriptors and
C     coefficients for the generalized delta rule to the STDOUT device.

      Parameter (NMax=15)
```

```
      Common /Netwrk/ NLayer,Nodes(0:3)
      Common /XferFn/ FMin,FMax,FGain,FRange
      Common /Trains/ Alpha,DAlpha,Beta,DBeta,DW(NMax,NMax,3),DB(NMax,3)

C ... Echo the training/testing information

      Write (*,7) NTrain,' = Number of training iterations'
      Write (*,7) NTest,' = Number of testing iterations'
      Write (*,7) NEpoch,' = Number of training epochs'
      Write (*,7) NPrint,' = Frequency of progress printouts'
      Write (*,8) Alpha,' = Initial delta rule training rate'
      Write (*,8) DAlpha,' = Training rate adjustment factor'
      Write (*,8) Beta,' = Initial delta rule momentum term'
      Write (*,8) DBeta,' = Momentum term adjustment factor'

C ... Echo the network dimensions

      Write (*,7) NLayer,' = Number of layers in network'
      Write (*,7) NIN,' = Number of inputs to network'

      Do 1 I = 1,NLayer-1
    1 Write (*,9) Nodes(I),' = Number of neurons in layer ',I

      Write (*,7) NOUT,' = Number of neurons in output layer'

C ... Echo the transfer function parameters

      Write (*,8) FMin,' = Transfer function minimum'
      Write (*,8) FMax,' = Transfer function maximum'
      Write (*,8) FGain,' = Transfer function gain'

      Return
    7 Format(I8,A)
    8 Format(F8.5,A)
    9 Format(I8,A,I1)
      End

C     ---------------------------------------------------------------

      Subroutine GETDATA (XIN,NIN,XOUT,NOUT,Count)

C     This FORTRAN routine reads in the input/output pattern pairs that
C     constitute the training set from an ASCII file. The max number of
C     input/output pattern pairs is given by NPairs and the max size of
C     each input or output pattern is given by NMax.

      Parameter (NMax=15,NPairs=1000)

      Integer Count
      Dimension XIN(NMax,NPairs),XOUT(NMax,NPairs)

      Open (1,File='training.data',Form='FORMATTED')

      Count = 1

C ... Read sets of input/output signals
```

```
    1 Read(1,*,End=2) (XOUT(K,Count),K=1,NOUT),(XIN(K,Count),K=1,NIN)
      Count = Count + 1
      If (Count.LE.NPairs) Go To 1

    2 Count = Count - 1

      Close (1)

      If (Count.EQ.0) STOP' No training data found in input file'

      Return
      End

C     ---------------------------------------------------------------

      SUBROUTINE ADVANCE (NUnit)

C     This FORTRAN routine reads through a sequential file to position
C     the file pointer at the end of the file.

    1 Read(NUnit,*,End=2)
      Go To 1

    2 Return
      End

C     ---------------------------------------------------------------

      SUBROUTINE RESET

C     This FORTRAN routine initializes the weight and bias arrays with
C     uniformly-distributed random numbers drawn between +5 and -5 and
C     initializes the weight and bias change arrays to zero.

      Parameter (NMax=15)

      Common /Netwrk/ NLayer,Nodes(0:3)
      Common /Weight/ W(NMax,NMax,3),B(NMax,3),Y(NMax,0:3)
      Common /Trains/ Alpha,DAlpha,Beta,DBeta,DW(NMax,NMax,3),DB(NMax,3)

C ... Initialize the weight and bias arrays to small random numbers

      DO 3 K = 1,NLayer
      DO 3 J = 1,Nodes(K)
      B(J,K) = 5.0 - 10.0*RANDOM(1)

      DO 3 I = 1,Nodes(K-1)
    3 W(I,J,K) = 5.0 - 10.0*RANDOM(2)

C ... Initialize the weight change and bias change arrays to zero

      DO 4 K = 1,NLayer
      DO 4 J = 1,Nodes(K)
      DB(J,K) = 0.0
      DO 4 I = 1,Nodes(K-1)
```

```
    4 DW(I,J,K) = 0.0

      Return
      End

C     ---------------------------------------------------------------

      SUBROUTINE ANN (XIN,NIN,YOUT,NOUT)

C     This FORTRAN routine implements a fully-connected neural network
C     with a variable number of layers and a variable number of neurons
C     in each layer. A sigmoidal transfer function with adjustable gain
C     and range is used in every neuron.

C     Subroutine arguments are an array (XIN) of inputs dimensioned by
C     the number of neurons (NIN) in the input layer and an array (YOUT)
C     of outputs dimensioned by the number of neurons (NOUT) in the
C     output layer.

      Parameter (NMax=15)

      Dimension XIN(NIN),YOUT(NOUT)
      Common /Netwrk/ NLayer,Nodes(0:3)
      Common /XferFn/ FMin,FMax,FGain,FRange
      Common /Weight/ W(NMax,NMax,3),B(NMax,3),Y(NMax,0:3)

C ... Copy network input array to 0th column of neuron output array

      Do 10 J = 1,NIN
   10 Y(J,0) = XIN(J)

C ... Propagate the input through all NLayer network layers

      Do 30 K = 1,NLayer
      Do 30 J = 1,Nodes(K)
      Sum = B(J,K)
      Do 20 I = 1,Nodes(K-1)
   20 Sum = Sum + Y(I,K-1)*W(I,J,K)
   30 Y(J,K) = FMin + FRange/(1.0+EXP(-FGain*Sum))

C ... Copy last column of neuron output array to network output array

      Do 40 J = 1,NOUT
   40 YOUT(J) = Y(J,NLayer)

      RETURN
      END

C     ---------------------------------------------------------------

      SUBROUTINE TRAIN (Error,NOUT)

C     This FORTRAN routine trains a fully-connected neural network with
C     a variable number of layers and a variable number of neurons in
C     each layer using the generalized delta rule with a momentum term.
C     Subroutine arguments are an output error array (Error) dimensioned
```

```
C     by the number of neurons (NOUT) in the output layer.

      Parameter (NMax=15)

      Dimension Error(NOUT),Delta(NMax,3)
      Common /Netwrk/ NLayer,Nodes(0:3)
      Common /XferFn/ FMin,FMax,FGain,FRange
      Common /Weight/ W(NMax,NMax,3),B(NMax,3),Y(NMax,0:3)
      Common /Trains/ Alpha,DAlpha,Beta,DBeta,DW(NMax,NMax,3),DB(NMax,3)

C ... Compute the deltas for the output layer

      Do 10 J = 1,NOUT
      Base = Y(J,NLayer) - FMin
      Slope = FGain*Base*(1.0 - Base/FRange)
   10 Delta(J,NLayer) = Error(J) * Slope

C ... Compute the deltas for the remaining (NLayer-1) layers

      Do 30 K = NLayer-1,1,-1
      Do 30 J = 1,Nodes(K)
      Sum = 0.0
      Base = Y(J,K) - FMin
      Slope = FGain*Base*(1.0 - Base/FRange)
      Do 20 I = 1,Nodes(K+1)
   20 Sum = Sum + Delta(I,K+1)*W(J,I,K+1)*Slope
   30 Delta(J,K) = Sum

C ... Compute the weight changes for all NLayer layers

      Do 40 K = 1,NLayer
      Do 40 J = 1,Nodes(K)
      DB(J,K) = Alpha*Delta(J,K) + Beta*DB(J,K)
      B(J,K) = B(J,K) + DB(J,K)
      Do 40 I = 1,Nodes(K-1)
      DW(I,J,K) = Alpha*Delta(J,K)*Y(I,K-1) + Beta*DW(I,J,K)
   40 W(I,J,K) = W(I,J,K) + DW(I,J,K)

      RETURN
      END

C     ------------------------------------------------------------------

      SUBROUTINE STATS (Error,NOUT,SqrErr)

C     This FORTRAN routine maintains accumulators for use in calculating
C     error statistics on the network output.

C     Subroutine arguments are an output error array (Error) dimensioned
C     by the number of neurons (NOUT) in the output layer and an updated
C     sum of the squared errors.

      Integer NOUT
      Real SqrErr,Error(NOUT)

      Do 1 J = 1,NOUT
```

```
          SqrErr = SqrErr + Error(J)*Error(J)
        1 Continue

          Return
          End

C         ------------------------------------------------------------------

          Real Function Random(Istrm)

*         This prime modulus multiplicative linear congruential generator is
*         based on Marse & Roberts' portable random-number generator Uniran:
*
*                     Z(I) = 630360016 * Z(I-1) * Mod(2**31 - 1)
*
*         Multiple streams are supported (100 in all), with the seeds spaced
*         100,000 apart.  Each time the function is invoked, the next random
*         variate in the specified stream (istrm) is returned.

          Integer B2e15,B2e16,Hi15,Hi31,Istrm,Low15,Lowprd
          Integer Modlus,Mult1,Mult2,Ovflow,Zi,Zrng(100)

*         Force saving of zrng between calls.

          Save Zrng

*         Define the constants.

          Data Mult1,Mult2 /24112,26143/
          Data B2e15,B2e16,Modlus /32768,65536,2147483647/

*         Set the default seeds for all 100 streams.

          Data Zrng /1973272912, 281629770,  20006270,1280689831,2096730329,
         &           1933576050, 913566091, 246780520,1363774876, 604901985,
         &           1511192140,1259851944, 824064364, 150493284, 242708531,
         &             75253171,1964472944,1202299975, 233217322,1911216000,
         &            726370533, 403498145, 993232223,1103205531, 762430696,
         &           1922803170,1385516923,  76271663, 413682397, 726466604,
         &            336157058,1432650381,1120463904, 595778810, 877722890,
         &           1046574445,  68911991,2088367019, 748545416, 622401386,
         &           2122378830, 640690903,1774806513,2132545692,2079249579,
         &             78130110, 852776735,1187867272,1351423507,1645973084,
         &           1997049139, 922510944,2045512870, 898585771, 243649545,
         &           1004818771, 773686062, 403188473, 372279877,1901633463,
         &            498067494,2087759558, 493157915, 597104727,1530940798,
         &           1814496276, 536444882,1663153658, 855503735,  67784357,
         &           1432404475, 619691088, 119025595, 880802310, 176192644,
         &           1116780070, 277854671,1366580350,1142483975,2026948561,
         &           1053920743, 786262391,1792203830,1494667770,1923011392,
         &           1433700034,1244184613,1147297105, 539712780,1545929719,
         &            190641742,1645390429, 264907697, 620389253,1502074852,
         &            927711160, 364849192,2049576050, 638580085, 547070247/

*         Generate the next random number.
```

```
 Zi     = Zrng(Istrm)

 Hi15   = Zi / B2e16
 Lowprd = (Zi - Hi15 * B2e16) * Mult1
 Low15  = Lowprd / B2e16
 Hi31   = Hi15 * Mult1 + Low15
 Ovflow = Hi31 / B2e15
 Zi     = (((Lowprd - Low15 * B2e16) - Modlus) +
&           (Hi31 - Ovflow * B2e15) * B2e16) + Ovflow

 If (Zi .Lt. 0) Zi = Zi + Modlus

 Hi15   = Zi / B2e16
 Lowprd = (Zi - Hi15 * B2e16) * Mult2
 Low15  = Lowprd / B2e16
 Hi31   = Hi15 * Mult2 + Low15
 Ovflow = Hi31 / B2e15
 Zi     = (((Lowprd - Low15 * B2e16) - Modlus) +
&           (Hi31 - Ovflow * B2e15) * B2e16) + Ovflow

 If (Zi .Lt. 0) Zi = Zi + Modlus

 Zrng(Istrm) = Zi

 Random = (2 * (Zi / 256) + 1) / 16777216.0

 Return
 End
```

REFERENCES

Anderson, J. A. (1972). "A simple neural network generating an interactive memory," *Mathematical Biosciences*, 14, 197-220. (Reprinted in Anderson and Rosenfeld, 1989.)

Anderson, J. A. (1988) "General Introduction," *Neurocomputing: foundations of research*, J. A. Anderson and Edward Rosenfeld, Eds., MIT, Cambridge, MA, xiii.

Anderson, J. A. and Rosenfeld, E. (1989) *Neurocomputing: Foundations of Research*, MIT, Cambridge, MA.

ASTM (1993) "Standard test method for deflections with a falling-weight-type impulse load device," *Annual Book of ASTM Standards*, 04.03, 566-568.

Barksdale, R. D. (1971) "Compressive stress pulse times in flexible pavements for use in dynamic testing," *Highway Research Record, 345*, Highway Research Board, Washington, DC.

Beaucham, R. E. (1993) "Evaluating Army airfields," *The Military Engineer* 83(541), 30.

Bentsen, R. A., Bush, A. J. III, and Harrison, J. A. (1989). "Evaluation of nondestructive test equipment for airfield pavements," Technical Report GL-89-3, U. S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

Block, H. D. (1962) "The Perceptron: a model for brain functioning, I," *Reviews of Modern Physics* 34, 123-135. (Reprinted in Anderson and Rosenfeld, 1989.)

Bohn, A., Ulidtz, P., Stubstad, R., and Sorensen, A. (1972). "Danish experiments with the French falling weight deflectometer," *Proceedings of the third international conference on the structural design of asphalt pavements*, University of Michigan, Ann Arbor, MI, 1119-1128.

Bouchon, M. (1981) "A simple method to calculate Green's functions for elastic layered media," *Bulletin of the Seismological Society of America*, 71, 959-971.

Burmister, D. M. (1943). "The theory of stresses and displacements in layered systems and applications to the design of airport runways," *Proceedings of the 23rd Annual Meeting of the Highway Research Board*, 23, 126-144.

Burmister, D. M. (1945a). "The general theory of stresses and displacements in layered soil systems, I," *Journal of Applied Physics* 16, 89-94.

Burmister, D. M. (1945b). "The general theory of stresses and displacements in layered soil systems, II," *Journal of Applied Physics* 16, 126-127.

Burmister, D. M. (1945c). "The general theory of stresses and displacements in layered soil systems, III," *Journal of Applied Physics* 16, 296-302.

Bush, A. J. III. (1980). "Nondestructive testing of light aircraft pavements, Phase II: development of the nondestructive evaluation methodology," Report FAA-RD-80-9-II. Federal Aviation Administration, Washington, DC.

Bush, A. J. III and Alexander, D. R. (1985) "Pavement evaluation using deflection basin measurements and layered theory," *Transportation Research Record 1022*, Transportation Research Board, Washington, DC, 16-29.

Bush, A. J. III, Brown, R. W., and Bailey, C. E. (1989) "Evaluation procedure for rigid airfield pavements," *Proceedings of the 4th International Conference on Concrete Pavement Design and Rehabilitation*, Purdue University, West Layfayette, IN, 419-429.

Chang, D. W., Kang, V. Y., Roesset, J. M., and Stokoe, K. H., II. (1992) "Effect of depth to bedrock on deflection basins obtained with Dynaflect and FWD tests," *Transportation Research Record 1355*, TRB, National Research Council, Washington, DC, 8-16.

Cooley, J. W. and Tukey, J. W. (1965) "An algorithm for machine calculation of complex Fourier series," *Math Computation*, 19, 297-301.

Cybenko, G. (1989) "Approximation by superposition of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, 2, 303-314.

Davies, T. G. and Mamlouk, M. S. (1985) "Theoretical response of multilayer pavement systems to dynamic nondestructive testing," *Transportation Research Record 1022*, TRB, National Research Council, Washington, DC, 1-7.

DeJong, D. L., Peutz, M. G. F., and Korswagen, A. R. (1973). "Computer program BISAR," Koninklijkel/Shell-Laboratorium, Amsterdam, the Netherlands.

Ewing, W. M., Jardetzky, W. S., and Press, F. (1957) *Elastic Waves in Layered Media*, McGraw-Hill, New York.

Foinquinos, R., Roesset, J. M., and Stokoe, K. H., II. (1993a) "FWD-DYN: A computer program for forward analysis and inversion of falling weight defection data," Research Report TX-94-1970-1F, Center for Transportation Research, The University of Texas at Austin, Austin, TX.

Foinquinos, R., Roesset, J. M., and Stokoe, K. H., II. (1993b) "Response of pavement systems to dynamic loads imposed by nondestructive tests," Preprint No. 940678, 73rd Annual Meeting of the Transportation Research Board, Washington, DC.

Freeman, J. A. and Skapura, D. M. (1991) *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley, Reading, MA, 104.

Hagiwara, M. (1992). "Theoretical derivation of momentum term in back-propagation," *Proceedings of the ICJNN92 Conference, 1*, IEEE, 682-686.

Haskell, N. A. (1953) "The dispersion of surface waves on multilayered media," *Bulletin of the Seismological Society of America*, 43, 17-34.

Hebb, D. O. (1949) *The Organization of Behavior*, Wiley and Sons, New York.

Hecht-Nielson, R. (1990) *Neurocomputing*, Addison-Wesley, Reading, MA.

Hoffman, M. S. and Thompson, M. R. (1982). "Comparative study of selected nondestructive testing devices," *Transportation Research Record 852*, Transportation Research Board, Washington, DC, 32-41.

Hopfield, J. J. (1982) "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, 79, 2554-2558. (Reprinted in Anderson and Rosenfeld, 1989.)

Hopfield, J. J. (1984). "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences*, 81, 3088-3092. (Reprinted in Anderson and Rosenfeld, 1989.)

Hornik, K., Stinchcombe, M., and White, H. (1989) "Multilayer feedforward networks are universal approximators," *Neural Networks* 2, 359-366.

Hornik, K., Stinchcombe, M., and White, H. (1990) "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks* 3, 551-560.

Houston, W. N., Mamlouk, M. S., and Perera, W. S. (1992) "Laboratory versus nondestructive testing for pavement design," *Journal of Transportation Engineering*, ASCE, 118(2), 207-222.

Hull, S. W. and Kausel, E. (1985) "Dynamic loads in layered halfspaces," *Engineering Mechanics in Civil Engineering*, ASCE, 1, 201-204.

Irwin, L. H. (1991) "User's guide to MODCOMP 3," CLRP Report Number 91-4, Cornell University, Ithaca, NY.

Irwin, L., Yang, W., and Stubstad, R. (1993) "Deflection reading accuracy and layer thickness accuracy in backcalculation of pavement layer moduli," *Nondestructive Testing of Pavements and Backcalculation of Moduli, ASTM STP 1026*, A. J. Bush III and G. Y. Baladi, Eds., ASTM, Philadelphia, 229-244.

Jones, R., Thrower, E. N. and Gatfield, E. N. (1968). "Surface wave method," *Proceeding of the Second International Conference on the Structural Design of Asphalt Pavements*, University of Michigan, Ann Arbor, MI, 404-421.

Kausel, E. (1981) "An explicit solution for the Green functions for dynamic loads in layered media," Research Report R81-13, Department of Civil Engineering, MIT, Cambridge, MA.

Kausel, E. and Peek, R. (1982) "Dynamic loads in the interior of a layered stratum: an explicit solution," *Bulletin of the Seismological Society of America*, 72, 1459-1481.

Kausel, E. and Roesset, J. M. (1981) "Stiffness matrices for layered soils," *Bulletin of the Seismological Society of America*, 71, 1743-1761.

Kohonen, T. (1972). "Correlation matrix memories," *IEEE Transactions on Computers*, C-21, 353-359. (Reprinted in Anderson and Rosenfeld, 1989.)

Lamb, H. (1904) "On the propagation of tremors over the surface of an elastic solid," *Philosophical Transactions of the Royal Society of London*, 203, 1-42.

Le Cun, Y. (1986). "Learning processes in an asymmetric threshold network," *Disordered Systems and Biological Organization*, E. Bienenstock, F. Fogelman Souli, and G. Weisbuch (Eds.), Springer, Berlin.

Letto, A. R. (1968). "A computer program for function optimization using pattern search and gradient summation techniques," Master Thesis, Texas A&M University, College Station, TX.

Lukanen, E. O. (1992) "Effects of buffers on falling weight deflectometer loadings and deflections," *Transportation Research Record 1355*, TRB, National Research Council, Washington, DC, 37-51.

Lysmer, J. and Waas, G. (1972) "Shear waves in plane infinite structures," *Journal of the Engineering Mechanics Division*, 98(EM1), 85-105.

Lytton, R. L. (1988) "Backcalculation of pavement layer properties," *Nondestructive Testing of Pavements and Backcalculation of Moduli, ASTM STP 1026*, A. J. Bush III and G. Y. Baladi, Eds., ASTM, Philadelphia, 7-38.

Matsuoka, K. (1992) "Noise injection into inputs in back-propagation learning," *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3), 436-440.

McCulloch, W. S. and Pitts, W. (1943) "A logical calculus of the ideas imminent in nervous activity," *Bulletin of Mathematical Biophysics*, 5, 115-133. (Reprinted in Anderson and Rosenfeld, 1989.)

Meier, R. W. and Rix, G. J. (1993) "An initial study of surface wave inversion using artificial neural networks," *Geotechnical Testing Journal*, 16(4).

Michelow, J. (1963) "Analysis of stresses and displacements in an n-layered elastic system under a load uniformly distributed on a circular area," California Research Corporation, Richmond, CA.

Miller, G. F. and Pursey, H. (1955) "On the partition of energy between elastic waves in a semi-infinite solid," *Proceedings of the Royal Society, London*, Series A, 233, 55-69.

Minsky, M. and Papert, S. (1969) *Perceptrons*, MIT Press, Cambridge, MA.

Nazarian, S. and Stokoe, K. H. II. (1989). "Nondestructive evaluation of pavements by surface wave method," *Nondestructive Testing of Pavements and Backcalculation of Moduli, ASTM STP 1026*, A. J. Bush III and G. Y. Baladi, Eds., ASTM, Philadelphia, 119-137.

Parker, D. B. (1986). "A comparison of algorithms for neuron-like cells," *Proceedings of the Second Annual Conference on Neural Networks for Computing*, J. Denker (Ed.), American Institute of Physics, New York, 327-332.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterline, W. T. (1989) *Numerical Recipes in FORTRAN*, Cambridge University Press, 422-497.

Rada, G. R., Richter, C. A., and Jordahl, P. (1994). "SHRP's layer moduli backcalculation procedure," *Nondestructive Testing of Pavements and Backcalculation of Moduli (Second Volume), ASTM STP 1198*, A. J. Bush III and G. Y. Baladi, Eds., ASTM, Philadelphia.

Rada, G. R., Richter, C. A., and Stephenson, P. J. (1992) "Layer moduli from deflection measurements: software selection and development of strategic highway research program's procedure for flexible pavements," *Transportation Research Record 1377*, TRB, National Research Council, Washington, DC, 77-87.

Richart, F. E., Jr., Hall, J. R., Jr., and Woods, R. D. (1970) *Vibrations of Soils and Foundations*, Prentice-Hall, Englewood Cliffs, NJ, 91.

Richter, C. A. and Rauhut, J. B. (1989). "SHRP plans for nondestructive deflection testing in the development of pavement performance prediction models," *Nondestructive Testing of Pavements and Backcalculation of Moduli, ASTM STP 1026*, A. J. Bush III and G. Y. Baladi, Eds., ASTM, Philadelphia, 556-562.

Roesset, J. M. and Shao, K-Y. (1985) "Dynamic Interpretation of Dynaflect and Falling Weight Deflectometer Tests," *Transportation Research Record 1070*, TRB, National Research Council, Washington, DC, 7-14.

Rohde, G. T. (1990) "The mechanistic analysis of pavement deflections on subgrades varying in stiffness with depth," Ph. D. dissertation, Texas A&M University, College Station, TX.

Rosenblatt, F. (1958) "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review* 65, 386-408. (Reprinted in Anderson and Rosenfeld, 1989.)

Rosenblatt, F. (1962) *Principles of Neurodynamics*, Spartan, New York.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). "Learning internal representation by error propagation," *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, D. E. Rumelhart and J. L. McClelland (Eds.), MIT Press, Cambridge, MA, 318-362. (Reprinted in Anderson and Rosenfeld, 1989.)

Sanchez-Salinero, I. (1987) "Analytical investigation of seismic methods used for engineering applications," Ph. D. dissertation, The University of Texas at Austin, Austin, TX.

Scrivner, F. H., Michalak, C. H., and Moore, W. M. (1973) "Calculation of the elastic moduli of a two-layer pavement system from measured surface deflection," *Highway Research Record 431*, Highway Research Board, Washington, DC, 12-22.

Sebaaly, B., Davies, T. G., and Mamlouk, M. S. (1985) "Dynamics of Falling Weight Deflectometer," *Journal of Transportation Engineering*, ASCE, 111(6), 618-632.

Sebaaly, B., Mamlouk, M. S., and Davies, T. G. (1985) "Dynamic Analysis of Falling Weight Deflectometer Data," *Transportation Research Record 1070*, TRB, National Research Council, Washington, DC, 63-68.

Smith, R. E. and Lytton, R. L. (1985). "Operating characteristics and user satisfaction of commercially available NDT equipment," *Transportation Research Record 1007*, Transportation Research Board, Washington, DC, 1-10.

Thomson, W. T. (1950) "Transmission of elastic waves through a stratified solid medium," *Journal of Applied Physics*, 21, 89-93.

Uzan, J., Lytton, R. L., and Germann, F. P. (1988). "General procedure for backcalculating layer moduli," *Nondestructive Testing of Pavements and Backcalculation of Moduli, ASTM STP 1026*, A. J. Bush III and G. Y. Baladi, Eds., ASTM, Philadelphia, 217-228.

Van Cauwelaert, F. J., Alexander, D. R., White, T. D., and Barker, W. R. (1988). "Multilayer elastic program for backcalculating layer moduli in pavement evaluation," *Nondestructive Testing of Pavements and Backcalculation of Moduli, ASTM STP 1026*, A. J. Bush III and G. Y. Baladi, Eds., ASTM, Philadelphia, 171-188.

Van Cauwelaert, F. J., Delaunois, F., and Beaudoint, L., (1986). "Computer programs for the determination of stresses and displacements in four-layered systems," WES Research Contract DAJA45-86-M-0483, U. S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

Walrond, G. and Christiansen, D. (1993) "Airfield pavement structural evaluations: current practice and future needs," *Airport Pavement Innovations-Theory to Practice*, ASCE, New York, 110-125.

Werbos, P. J. (1974). "Beyond regression: new tools for prediction and analysis in the behavioral sciences," Ph. D. dissertation, Harvard University, Cambridge, MA.

Widrow, B. and Hoff, M. E. (1960) "Adaptive switching circuits," *1960 IRE WESCON Convention Record*, 4, 96-104. (Reprinted in Anderson and Rosenfeld, 1989.)

Yih Hou, T. (1977). "Evaluation of layered material properties from measured surface deflections," Ph. D. dissertation, University of Utah.

Yoder, E. J., and Witczak, M. W. (1975) *Principles of Pavement Design*, John Wiley and Sons, New York, 129.

VITA

Roger William Meier was born in Summit, New Jersey on June 6, 1957, the son of Charles A. and Daphne M. Meier. He graduated from Governor Livingston Regional High School in Berkeley Heights, New Jersey in 1975 and enrolled at Virginia Polytechnic Institute and State University in Blacksburg, Virginia to study Civil Engineering. Mr. Meier received a Bachelor of Science in Civil Engineering from Virginia Tech in 1979 and enrolled at the University of Colorado in Boulder, Colorado to pursue his Master's degree. He was awarded a Master of Science in Civil Engineering from the University of Colorado in 1983. He has been employed since February 1983 by the U. S. Army Engineer Waterways Experiment Station in Vicksburg, Mississippi as a Research Civil Engineer. In 1990, Mr. Meier enrolled at Georgia Institute of Technology to pursue his doctorate. While at Georgia Tech, he met and married the former Ann Raney Jay from Atlanta, Georgia.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1995	Final Report

4. TITLE AND SUBTITLE
Backcalculation of Flexible Pavement Moduli from Falling Weight Deflectometer Data Using Artificial Neural Networks

5. FUNDING NUMBERS

6. AUTHOR(S)
Roger W. Meier

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Engineer Waterways Experiment Station
3909 Halls Ferry Road, Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER
Technical Report GL-95-3

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Available from National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The goal of this research was to develop a method for backcalculating pavement layer moduli from FWD data in real time. This was accomplished by training an artificial neural network to approximate the backcalculation function using large volumes of synthetic test data generated by static and dynamic pavement response models. One neural network was trained using synthetic test data generated by the same static, layered-elastic model used in the conventional backcalculation program WESDEF. That neural network was shown to be just as accurate but 2500 times faster. The same neural network was subsequently retrained using data generated by an elastodynamic model of the FWD test. The dynamic analysis provides a much better approximation of the actual test conditions and avoids problems inherent in the static analysis. Based on the amounts of time needed to create the static and dynamic training sets, a conventional program would likely run 20 times slower if it employed the dynamic model. The processing time of the neural network, on the other hand, is unchanged because it was simply retrained using different data. These artificial neural networks provide the real-time backcalculation capabilities needed for more thorough, more frequent, and more cost-effective pavement evaluations. Furthermore, they permit the use of more-realistic models, which can increase the accuracy of the backcalculated moduli.

14. SUBJECT TERMS
Pavement analysis, Backcalculation, Falling weight deflectometer, Artificial neural networks

15. NUMBER OF PAGES
239

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	